Universidade Federal de Santa Catarina

Programa de Pós-graduação em

Engenharia de Produção

UM COMPONENTE COMPUTACIONAL PARA AUXILIAR O DESENVOLVIMENTO DE UMA ASSINATURA DIGITAL NO SISTEMA DE INFORMAÇÕES PROCESSUAIS

Dissertação de Mestrado

Paulo de Tarso Deliberador

Florianópolis

2004

Universidade Federal de Santa Catarina

Programa de Pós-graduação em

Engenharia de Produção

UM COMPONENTE COMPUTACIONAL PARA AUXILIAR O DESENVOLVIMENTO DE UMA ASSINATURA DIGITAL NO SISTEMA DE INFORMAÇÕES PROCESSUAIS

Paulo de Tarso Deliberador

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Santa Catarina como requisito parcial para obtenção do título de Mestre em Engenharia de Produção

Orientador: Prof. Milton Luiz Horn Vieira, Dr.

Florianópolis

2004

Paulo de Tarso Deliberador

UM COMPONENTE COMPUTACIONAL PARA AUXILIAR O DESENVOLVIMENTO DE UMA ASSINATURA DIGITAL NO SISTEMA DE INFORMAÇÕES PROCESSUAIS

Esta dissertação foi julgada e aprovada para a obtenção do título de **Mestre em Engenharia de Produção** no **Programa de Pós-Graduação em Engenharia de Produção** da Universidade Federal de Santa Catarina

Florianópolis, 01 de Dezembro de 2004

Prof. Edson Pacheco. Paladini, Dr.
Coordenador do Curso

BANCA EXAMINADORA

Profº. Milton Luiz Horn Vieira, Dr.
Universidade Federal de Santa Catarina
Orientador

Profº. Mauro José dos Santos, Dr.
Universidade Federal de Santa Catarina

Profª. Valdete Teixeira da Silva , Dra.
Universidade Federal de Santa Catarina

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dado esta oportunidade e por nunca ter me abandonado em nenhum momento.

Ao meu pai Francisco e minha mãe Cecília pela dedicação e carinho, e principalmente pelo exemplo de vida que pretendo copiar e seguir por toda a minha vida que mesmo com seus problemas pessoais nunca deixaram de me incentivar e apoiar.

Aos meus irmãos e irmãs que sempre acreditaram no meu potencial e me apoiaram em minhas decisões.

Ao professor Dr. Milton Luiz Horn Vieira pela sua atenção, compreensão e experiência transmitida no decorrer destes anos,sendo fundamental para a realização deste trabalho.

A minha namorada Mellina pela compreensão sobre a minha ausência, e mesmo distante nunca deixou de acreditar em mim.

Ao Góis e a Eliana, que acreditaram no meu potencial e me indicaram para a realização desta pesquisa.

Aos meus colegas e amigos pelo incentivo, alegria e momentos de descontração, sendo fundamental para a reflexão sobre os problemas encontrados no decorrer desta pesquisa.

A todos que direta ou indiretamente contribuíram para a realização desta conquista, dedico aqui todo o meu agradecimento e carinho.

“A mente que se abre a uma nova idéia
jamais voltará ao seu tamanho original.”

Albert Einstein

RESUMO

DELIBERADOR, Paulo de Tarso. **Um componente computacional para auxiliar o desenvolvimento de uma assinatura digital no Sistema de Informações Processuais.** 2004. 198 f. Dissertação (Mestrado em Engenharia de Produção) – Programa de Pós-Graduação em Engenharia de Produção, UFSC, Florianópolis.

Este trabalho apresenta conceitos relacionados à assinatura digital, e a necessidade de garantir a confiabilidade e autenticidade de informações em ambientes computacionais. O intuito da assinatura digital é fazer com que os documentos digitais tenham o mesmo valor legal do que um documento escrito e assinado de maneira comum, com a finalidade de garantir a autoria e autenticar a origem dos dados contidos no documento. Esta dissertação tem como objetivo apresentar um componente computacional para auxiliar programadores e analistas de sistemas, facilitando a aplicação da técnica de assinatura digital na integração do Sistema de Informações Processuais (SIP), desenvolvido pelo DesignLAB para aplicação na Justiça Trabalhista Brasileira em parceria com o Departamento de Expressão Gráfica da Universidade Federal de Santa Catarina. Com o componente computacional proposto, pretende-se facilitar a implementação da assinatura digital, aperfeiçoando a segurança do sistema atual, garantindo a integridade e autenticação informatizada dos documentos criados ou alterados pelos usuários do sistema.

PALAVRAS-CHAVE: Criptografia, Algoritmos Criptográficos, Assinatura Digital, Segurança de Dados, Assinatura Digital em Java.

ABSTRACT

DELIBERADOR, Paulo de Tarso. **Um componente computacional para auxiliar o desenvolvimento de uma assinatura digital no Sistema de Informações Processuais.** 2004. 198 f. Dissertação (Mestrado em Engenharia de Produção) – Programa de Pós-Graduação em Engenharia de Produção, UFSC, Florianópolis.

This work presents related concepts to the digital signature, and the need to guarantee the reliability and authenticity of information in computing environments. The intention of the digital signature is to make the digital documents have the same legal value that a document writing and signed in a common way, with the purpose of guarantee the authorship and to authenticate the origin of the data contained in the document. This dissertation has the purpose to present a computing component to aid programmers and systems administrators in facilitating the application of the technique for digital signature and the integration of the Procedural Information System (PIS), has developed by the DesignLAB for application in the Labor Court of Justice in partnership with the Graphic Design Department of Santa Catarina Federal University. With the computing component proposed, intends to facilitate the implementation of the digital signature, improving the security of the current system, guaranteeing the integrity and computerized the documents' authentication created or altered by the system users.

KEY-WORDS: Cryptography, Cryptographic Algorithms, Digital Signature, Data Security, Java Digital Signature.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS

Nomes e Siglas

ABA -	American Bar Association
AC Raiz -	Autoridade Certificadora Raiz
AES -	Padrão de Criptografia Avançada - Advanced Encryption Standard
AGP -	Autoridade de Gerência de Políticas
ANSI -	National Institute of Standards in Technology
API -	Apllication Programming Interface
CA -	Certificate Authority - Autoridade Certificadora
CPP -	Criptography Package Provider
CRLs	Certification Revocation Lists - Lista de Revogação de Certificados
DES -	Digital Encryption Standard - Padrão de Criptografia Digital
DH -	Diffie, Hellman
DSA -	Digital Signature Algorithm
DSG -	Digital Signature Guidelines
DSS -	Digital Signature Standard
EC -	Elliptic Curve
ECDH -	Elliptic Curve Diffie Hellman
ECDSA -	Elliptic Curve Digital Signature Algorithm
EDI -	Eletronic Data Interchange
E-SIGN	Eletronic Signatures in Global And National Commerce - Assinaturas Eletrônicas no Comércio Nacional e Global
FIPS -	Federal Information Processing Standard
HMAC -	Hashed Message Authentication Code - Códigos de Autenticação de Mensagem
IBM -	Information Business Machine
ICP -	Infra-Estrutura de Chave Pública
ICP-Gov -	Infra-Estrutura de Chaves Públicas do Poder Executivo Federal
ICP-OAB	Infra-Estrutura de Chaves Públicas da Ordem dos Advogados
IDEA -	International Data Encryption Algorithm - Algoritmo de Criptografia de Dados Internacionais
IETF -	Internet Engineering Task Force
IP-	Internet Protocol

IPSec -	Internet Protocol Security
ISSO -	International Organization for Standardization
JCA -	Java Cryptography Architecture – Arquitetura de Criptografia Java
JCE -	Java Cryptography Extension – Extensão da Criptografia Java
JDK -	Java Development Kit
JSDK	Java Software Development Kit
JVM -	Java Virtual Machine – Máquina Virtual Java
LCR -	Lista de Certificados Revogados
LDAP -	Lightweight Directory Access Protocol
M.I.T -	Massachussets Institute of Technology
MAC	Código de Autenticação de Mensagem - Message Authentication Code
MD -	Message Digest
Message Digest -	Resumo de mensagem
NIST -	National Institute of Standards and Technology
NSA -	National Security Agency
OAB-SP -	Ordem dos Advogados do Brasil – São Paulo
OSI -	Open System Interconnection
PGP -	Pretty Good Privacy
PKC -	Public Key Cryptografic
PKI -	Public Key Infrastructure - Infra Estrutura da Chave pública
PRNG -	Pseudo-Randon Number Generators – Gerador de Números Pseudo-Aleatórios
RA -	Registration Authorities - Autoridades de Registro
RAM -	Random Access Memory
RC	Ron's Cipher
RSA -	Rivest-Shamir-Adleman
S/MIME -	Secure Multipurpose Internet Mail Extensions
SHA -	Secure Hash Algorithm
SSL -	Secure Socket Layer
SPB -	Sistema de Pagamentos Brasileiros
TCP -	Transmission Control Protocol
TCP / IP -	Transmission Control Protocol / Internet Protocol
TRT -	Tribunal Regional do Trabalho
Triple DES -	Triple Digital Encryption Standard – Padrão de Encriptação Digital Tripla
TSA -	Time Stamping Authority - Autoridade Registradora de data/hora
UNCITRAL -	United Nations Commission on International Trade Law

UNCITRAL -	United Nations Commission on International Trade Laic - Comissão das Nações Unidas sobre o Direito do Comércio Internacional
Usenet -	Unix User Network
X.509	Certificado de chave pública criada pela International Telecommunications Union

SUMÁRIO

1 INTRODUÇÃO

Com o avanço tecnológico da sociedade moderna, a *internet* está se consolidando como um instrumento fundamental para comunicação empresarial e pessoal, aumentando cada vez mais a preocupação com a segurança na troca de informações, podendo esta segurança ser a garantia de integridade da informação, autenticação de usuários, ou até mesmo a privacidade dos dados trafegados em uma rede de computador. A solução apontada pelos especialistas em segurança de dados é a implementação da denominada “assinatura digital”. (BURNETT E PAINE, 2002).

A assinatura digital poderá ser um meio efetivo que garantirá as propriedades de integridade e autenticação dos dados transmitidos na rede ou armazenados em meio computacionais. O processo de implementação de uma assinatura digital pode ser garantido por meio de tecnologias baseadas em um sistema criptográfico assimétrico, composto de um algoritmo ou uma série de algoritmos que irão gerar um par de chaves, sendo uma privada – usada para assinar um documento – e uma chave pública – usada para verificar a validade da assinatura (SCHNEIER, 2001).

A proposta do presente trabalho é desenvolver e apresentar um componente computacional em forma de pacote (*package)*, contendo classes desenvolvidas utilizando a linguagem de programação Java, que auxiliará desenvolvedores de sistemas de computador a integrar a técnica da assinatura digital ao Sistema de Informações Processuais (SIP), desenvolvido pelo DesignLAB para aplicação na Justiça Trabalhista Brasileira em parceria com o Departamento de Expressão Gráfica da Universidade Federal de Santa Catarina.

Com o desenvolvimento deste componente, pretende-se facilitar a produção de aplicações baseadas na assinatura digital para os módulos do sistema atualmente em uso, aperfeiçoando a segurança, garantindo a integridade e a autenticação informatizada de todo e qualquer trâmite processual criado ou alterado pelos usuários.

1.1 Origem do Trabalho

A origem deste trabalho encontra-se na parceria estabelecida entre o DesignLab (Laboratório de design / UFSC) e a Justiça Trabalhista Brasileira no desenvolvimento do aplicativo denominado Sistema de Informações Processuais (SIP), e reside na possibilidade de aperfeiçoar a segurança do sistema, desenvolvendo um componente computacional em forma de pacote (*package)* para auxiliar analistas de sistemas de programadores a produzir aplicações baseadas na tecnologia da assinatura digital, com o intuito de integrar este componente ao Módulo de Segurança do sistema, possibilitando que seus usuários possam assinar digitalmente todos e quaisquer trâmites processuais e posteriormente, realizar a verificação de validade da assinatura inserida.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um componente computacional em forma de pacote (*package)*, utilizando a linguagem de programação Java, que auxiliará desenvolvedores de sistemas de computador a integrar a técnica da assinatura digital ao Sistema de Informações Processuais (SIP).

1.2.2 Objetivo Específico

- Realizar uma Revisão Bibliográfica sobre os conceitos relevantes ao projeto, apresentando os principais algoritmos de criptografia, com finalidade de formar um suporte teórico sobre o funcionamento da assinatura digital.
- Identificar as principais características e facilidades que a Linguagem de Programação Java oferece na implementação de um processo de assinatura digital.
- Verificar a viabilidade de aplicação dos recursos da assinatura digital utilizando o componente computacional no Sistema de Informações Processuais (SIP).
- Explicar os recursos do componente computacional desenvolvido e as principais formas de utilização do mesmo.

1.3 Procedimentos Metodológicos

Para garantir a autoria e a integridade de documentos digitais, verificou-se a necessidade de se implementar um componente computacional para auxiliar programadores e analistas de sistemas a produzirem aplicativos baseados nas técnicas de assinatura digital, para incorporar o Sistema de Informação Processual. A partir deste propósito, foi realizada uma pesquisa que utilizou recursos bibliográficos diversos, tais como livros, revistas e pesquisas na *Internet* (artigos, tutoriais), com o objetivo de apresentar os principais aspectos técnicos, práticos e legais relacionados à assinatura digital.

Através destas pesquisas, constatou-se que a assinatura digital está embasada na adoção da criptografia assimétrica, sendo esta privilegiada na pesquisa quanto a seus aspectos técnicos, com a apresentação dos algoritmos mais utilizados para gerar chaves criptográficas em assinaturas digitais.

Quanto ao objetivo de implementar o componente computacional utilizando a técnica da assinatura digital, foi necessário efetivar um levantamento para identificar as funcionalidades do aplicativo “Sistema de Informação Processual (SIP)”. O cumprimento desta etapa foi realizado através de um acompanhamento diário do sistema em questão por um período de um ano, aproximadamente. Após este período, foram realizadas várias reuniões com os técnicos responsáveis pelo desenvolvimento do SIP, nas quais foram analisados os seguintes tópicos:

- A disponibilidade de recursos físicos.
- A viabilidade de implementação de aplicativos utilizando a técnica de assinatura digital.
- O levantamento do ambiente de aplicação.

Após a definição do ambiente de aplicação, um estudo sobre as principais características da Linguagem de Programação Java foi realizado. Também realizou-se um estudo sobre os principais algoritmos utilizados em todo o processo de assinatura digital, sendo possível a implementação destes algoritmos utilizando a linguagem de programação Java.

A partir do resultado deste estudo, constatou-se a possibilidade da implementação de um componente computacional capaz de auxiliar na produção de aplicativos baseados nas técnicas de assinatura digital, tendo como princípio básico a geração do par de chaves, sendo uma chave privada e uma chave pública; assinar um documento digital e verificá-lo posteriormente.

1.4 Descrição e Organização dos Capítulos

Este trabalho está estruturado em cinco capítulos distribuídos da seguinte forma:

Capítulo 1 – Introdução

Este capítulo descreve o contexto deste trabalho, com a Introdução do tema da pesquisa, na qual estão apresentados os seguintes itens: origem do trabalho, os

objetivos, procedimentos metodológicos e as limitações encontradas durante a realização deste trabalho.

Capítulo 2 – Infra-estrutura da assinatura digital

Neste capítulo é apresentada a infra-estrutura da assinatura digital através da revisão da literatura sobre o assunto, constituindo-se como fundamentação teórica para a criação de um processo de assinatura digital. São apresentados os conceitos básicos sobre criptografia, os principais algoritmos criptográficos, conceitos e técnicas aplicadas a uma assinatura digital e a certificados digitais, além de um breve histórico sobre as abordagens legislativas relativas ao tema.

Capítulo 3 – Modelagem da assinatura digital em Java

Este capítulo trata das principais características e das facilidades que a Linguagem Java oferece para se implementar um processo de assinatura digital. Descreve-se a implementação do componente computacional, utilizando as facilidades da linguagem em todas as etapas de criação de uma assinatura digital: geração de chaves de assinatura, geração da assinatura e a verificação.

Capítulo 4 – Aplicação dos recursos de assinatura digital, utilizando o componente computacional desenvolvido para o sistema de informações processuais.

Neste capítulo realizou-se a descrição do Sistema de Informações Processuais (SIP), seus principais objetivos, características e funcionalidades de cada módulo, assim como a aplicação do componente computacional neste sistema. Este capítulo também explica as principais formas de utilização do componente computacional através de exemplos, utilizando a sintaxe da linguagem de programação Java e o detalhamento das classes contidas no pacote deste componente.

Capítulo 5 – Conclusão

Apresenta as conclusões do trabalho, bem como as contribuições e recomendações para trabalhos futuros. E, posteriormente, são listadas as Referências, a Bibliografia consultada e os Anexos.

2 INFRA-ESTRUTURA DA ASSINATURA DIGITAL

2.1 Criptografia

Formada a partir da concatenação dos termos gregos *kryptos* (escondido, oculto) e *graphé* (grafia, escrita), a criptografia é a ciência que torna possível a comunicação mais segura entre dois agentes, sobre um canal aberto, convertendo dados legíveis em algo sem sentido, com a capacidade de recuperar os dados originais a partir destes.

Segundo Burnett e Paine (2002), existem alguns termos oficiais relacionados à criptografia. Estes termos serão muito utilizados no decorrer desta dissertação, como por exemplo: o processo que converte informações sigilosas em algo sem sentido, costuma-se utilizar os termos (*encripta, codifica, criptografa, cifra*), e para tornar as informações sigilosas legíveis novamente, utiliza-se os termos (*decripta, decodifica, decriptografa, decifra*). O estudo sobre a quebra de sistemas de criptografia é conhecido como *análise criptográfica* e os profissionais que utilizam esta técnica com o intuito de descobrir e divulgar as fraquezas dos algoritmos de criptografia são conhecidos como *criptoanalistas*, ao contrário dos *invasores* que dificilmente divulgam as possíveis fraquezas encontradas nestes algoritmos. Já os *criptógrafos*, são os profissionais que desenvolvem os sistemas de criptografia.

Apesar de muitos acreditarem que esta ciência foi desenvolvida para uso militar durante a Primeira e a Segunda grande guerra, a criptografia existe há milhares de anos. Há evidências de que ela já se encontrava presente no sistema de escrita hieroglífica dos egípcios e em documentos de generais gregos e romanos, porém somente no início dos anos 70 que surgiu como área de investigação acadêmica de conhecimento generalizado.

Para Terada (2000, p.16), a Criptografia pode ser descrita como:

> a ciência que estuda a transformação de dados de maneira a torná-los incompreensíveis sem o conhecimento apropriado para a sua tradução, tornando os conteúdos secretos, evitando riscos internos e externos que venham a ocorrer durante o trajeto dos dados enviados, que são convertidos em um código que só poderão ser traduzidos por quem possuir a "chave" secreta, enquanto que a Criptoanálise executa o processo inverso, sendo a ciência que estuda a decifração, tornando o código compreensível.

Nos dias atuais, a criptografia e sua utilização nos sistemas de informações ganham cada vez mais importância, pois com as redes abertas de informações como a *internet*, por exemplo, é cada vez maior a necessidade da utilização desta ciência para que pessoas mal intencionadas não utilizem seus dados em beneficio próprio. Desta forma, a criptografia poderá ser utilizada para codificar dados e mensagens, antes de serem enviados pelas redes abertas de informações, e ainda que estes dados sejam interceptados, dificilmente seriam legíveis pelos interceptadores. Para tanto, são utilizadas funções matemáticas e uma senha especial chamada "chave".

Esta chave é de fundamental importância, pois sem a utilização dela, o interceptador da mensagem só teria que saber qual foi o algoritmo empregado na criptografia e utilizar o mesmo algoritmo para decriptografar a mensagem (OAKS, 1999). Apesar de não ser algo simples de se fazer, existem pessoas especializadas em descobrir o algoritmo empregado na criptografia. Desta forma, a utilização de uma chave se torna mais difícil para decriptografar a mensagem supostamente interceptada. Para um melhor entendimento da utilização desta chave, imaginem uma porta muito resistente em uma residência. Nesta porta é colocada uma fechadura que qualquer chave possa abrir. Para quem passa do lado de fora da casa, a porta esta fechada e a casa aparentemente segura, porém, qualquer pessoa pode abrí-la simplesmente, girando a fechadura.

A figura 1 a seguir, mostra esquematicamente, como se dá o processo criptográfico na transmissão de dados e/ou informações.

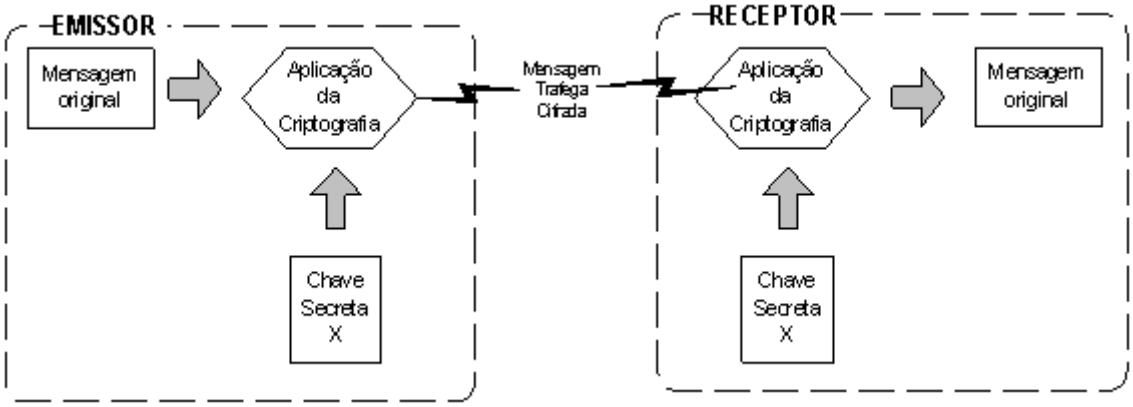

FIGURA 1 – Ilustração de um processo simples de criptografia no envio da mensagem.
Fonte: Volpi (2000, p.7).

Segundo Garfinkel e Spafford (1999, p.209) as quatro palavras-chave usadas para descrever todas as diferentes funções que a criptografia desempenha nos sistemas modernos de informações são:

- **Confidencialidade ou Privacidade:** utilizada para embaralhar as informações enviadas por tele-transmissão em canais abertos e armazenadas em um servidor, de forma que os invasores não possam acessar o conteúdo dos dados. Refere-se à proteção de informações privadas (confidenciais ou não) contra o uso ou agregação imprópria. Impede que pessoas não autorizadas tenham acesso ao conteúdo da mensagem ou parte dela, e que, posteriormente, essas pessoas possam utilizar os dados obtidos de forma desautorizada para interferir sobre o todo. Sua finalidade é garantir que apenas a origem e o destino tenham conhecimento do exato teor das informações;

- **Autenticação:** garante a identidade de quem está enviando a mensagem. O receptor da mensagem pode verificar a identidade da pessoa que a assinou. Pode ser implementada a partir de um mecanismo de senhas ou de assinatura digital;

- **Integridade:** cuida para que o conteúdo da mensagem não seja modificado em trânsito. Os dados recebem um tratamento que os transformam em caracteres não legíveis, garantindo que qualquer

receptor, que não o destinatário dos dados, fique impedido de modificá-los ou deles se apropriar. Dessa forma, a informação transmitida em um ponto é a mesma recebida em outro.

- **Não – repúdio:** tem a função de fazer com que o autor de uma mensagem não possa negar falsamente que a tenha enviado. Assim, se uma entidade enviou uma mensagem à outra entidade, não podem negar a autoria ou o recebimento. Usa-se, para tanto, procedimentos criptográficos, aliados a técnicas de assinatura digital.

2.1.1 Cifras e Chaves

Uma cifra é um algoritmo criptográfico, uma função matemática que efetua as transformações entre o texto limpo e o criptograma. Para um melhor entendimento, a palavra “algoritmo” é um termo utilizado para nomear uma “receita” de procedimentos, passo a passo. Pode-se dizer que um algoritmo é uma lista de instruções, que deverão ser executadas em uma determinada ordem, ou, ainda, que é uma lista rígida de comandos a ser seguida; essa lista pode conter uma série de perguntas e, dependendo das respostas, descreve os passos apropriados a serem seguidos (BURNETT E PAINE, 2002).

Tradicionalmente, os detalhes das cifras eram secretos, residindo neste principio a sua segurança, porém na criptografia moderna, a segurança de uma cifra não é adquirida simplesmente tornando secreto o seu funcionamento.

Na criptografia computadorizada, os algoritmos são, às vezes, operações matemáticas complexas e, em outros casos, apenas manipulações de bits. Existem vários algoritmos de criptografia e cada um tem sua própria lista de comandos ou passos. É possível haver um programa que implemente um algoritmo de criptografia, procedendo pela conversão dos dados recebidos em algo sem sentido; isso exige, necessariamente, o uso de uma chave (TERADA, 2000).

As chaves são elementos fundamentais que interagem com os algoritmos para cifrar/decifrar as mensagens. O algoritmo realiza seus passos utilizando a

chave para alterar o texto normal e convertê-lo em texto cifrado. Para decifrar o texto é necessário inserir a mesma chave para que o algoritmo torne o texto legível novamente, conforme ilustrado na figura 2, abaixo:

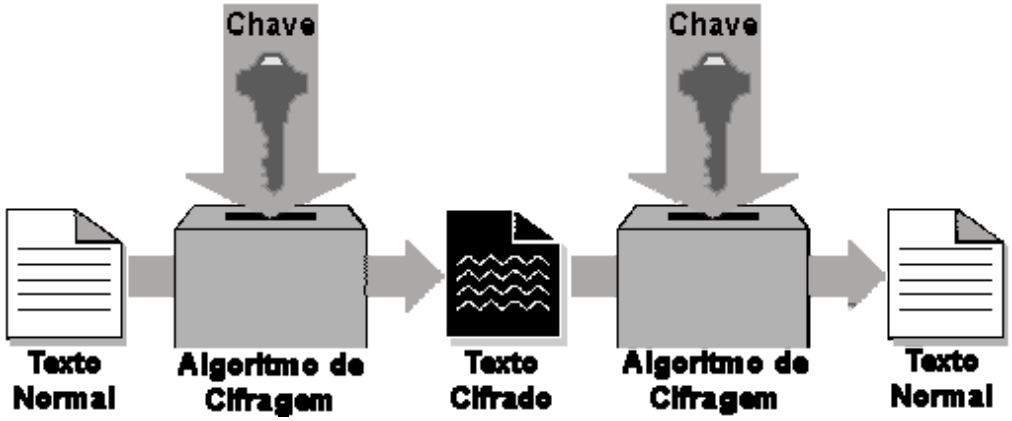

FIGURA 2 – Ilustração de um processo para cifrar e decifrar utilizando chaves.
Fonte: GARFINKEL, Simson; SPAFFORD, Gene (1999, p.189.

Segundo Burnett e Paine (2002 p. 10), a criptografia não garante a segurança de dados, nem resolverá todos os problemas de segurança, não é a prova de falhas, podendo ser quebrada, sobretudo se for implementada incorretamente. Porém, pode-se dificultar muito a quebra de segurança, com a utilização das chaves nos algoritmos de criptografia.

As chaves de criptografia são similares às senhas de acesso a bancos e a sistema de acesso a computadores; a utilização da senha correta torna possível ter acesso aos serviços, em caso contrário, o acesso é negado. No caso da criptografia, o uso de chaves relaciona-se com o acesso ou não à informação cifrada. O usuário deve usar a chave correta para poder decifrar as mensagens.

Assim como as senhas, as chaves na criptografia também possuem diferentes tamanhos, pois seu grau de segurança está relacionado à sua extensão, ou seja, quanto maior for a senha de um usuário, maior será o grau de confidencialidade da mensagem. Na criptografia, as chaves são longas seqüências de bits. Assim, uma chave de três dígitos oferecerá oito ($2^3 = 8$) possíveis valores para a chave (LYNCH E LUNDQUIST, 1996).

Há, basicamente, dois tipos de criptografia em relação ao uso de chaves. Quando é utilizada a mesma chave, tanto para cifrar quanto para decifrar uma mensagem, o método adotado é chamado de sistema de criptografia por chave simétrica ou chave secreta. Caso se utilizem chaves diferentes para cifrar e decifrar

uma mensagem, este procedimento é nomeado de sistema de chaves assimétricas ou chave pública.

2.1.2 Criptografia Simétrica

Conhecido também como Criptografia de Chave secreta, ou também de Criptografia Convencional. O principio básico deste conceito é a utilização de uma chave secreta, que o emissor utiliza para codificar um documento, e posteriormente, o destinatário a utilizará para decodificar o documento. Nesse sistema, tanto o emissor quanto o receptor da mensagem cifrada devem compartilhar a mesma chave, que deve ser mantida em segredo por questão de segurança, pois, através desta chave e do algoritmo de cifragem utilizado para criptografar os dados, torna-se possível decriptografá-los.

Como citado no parágrafo anterior, a Criptografia Simétrica está baseada em uma única chave. A figura 3 ilustra este processo de forma clara, mostrando que a mesma chave que cifra uma mensagem é utilizada para decifrá-la posteriormente.

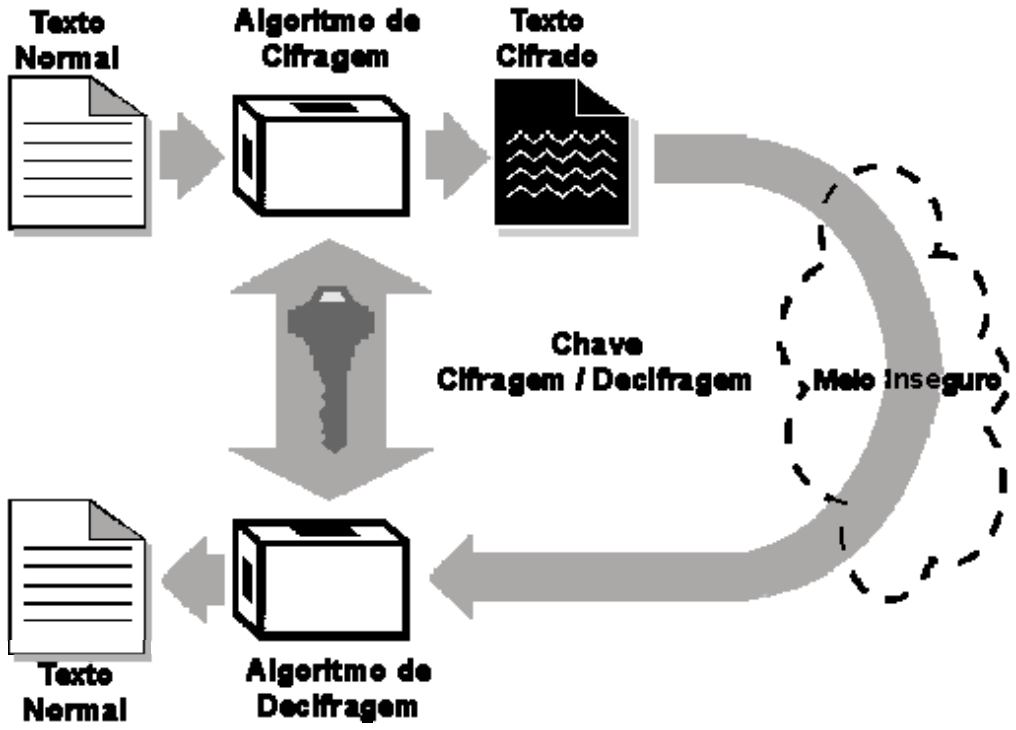

FIGURA 3 – Ilustração de um processo de criptografia por chave secreta
Fonte: GARFINKEL, Simson; SPAFFORD, Gene (1999, p.197).

Segundo Burnett e Paine (2002 p. 18), em um sistema criptográfico de chave simétrica, a chave é apenas um número qualquer, contando que tenha um tamanho correto e que seja selecionado aleatoriamente. Este pode ser um aspecto desfavorável para este tipo de chave, pois imaginando que um invasor queira ler os dados criptografados, sendo capaz de descobrir qual é a chave, ele pode decriptografar os dados, tornado-os legíveis novamente. Um método utilizado para descobrir esta chave é chamado de “ataque de força bruta”, onde um invasor tenta todas as possíveis chaves até que esta seja identificada. Digamos que a chave seja um número entre 0 e 100.000.000.000 (cem bilhões), o invasor pega o texto criptografado e alimenta o algoritmo de cifragem junto com a “alegada chave” de “0”. O algoritmo realiza seu trabalho e gera um resultado, caso o resultado gerado seja um texto sem sentido, “0” não é a chave verdadeira, então o invasor tenta com a chave 1 depois 2,3,4 e assim por diante, até que a chave correta seja encontrada. É importante ressaltar que o algoritmo simplesmente realiza os passos, independente da entrada. O algoritmo não tem nenhuma maneira de saber se o resultado produzido é satisfatório ou não, sendo sempre necessário examinar se o resultado gerado é um texto válido.

Outro aspecto desfavorável desse método é o processo de distribuição de chaves. Segundo Stallings (1999, p.141), o maior problema deste tipo de sistema é garantir que o emissor e o destinatário de uma mensagem cifrada pelo algoritmo – e somente eles – possam conhecer a chave secreta ora em uso. Além disso, existe a necessidade de transmitir a chave secreta por um meio seguro, separadamente da mensagem, e de efetivar combinações prévias sobre futuras alterações daquela.

Isto requer a existência de um método pelo qual as duas partes possam se comunicar de modo seguro, capaz de garantir a segurança do sigilo. Caso esse passo do processo sofresse quebra de sigilo, nessa forma de distribuição de chaves, qualquer interceptador poderia ter acesso tanto à mensagem, quanto à chave secreta. Um outro problema que ocorreria nesse processo seria o caso de três pessoas – A, B e C – que quisessem se comunicar utilizando chaves secretas. Seriam necessárias 3 (três) chaves: uma compartilhada entre A e B, outra entre A e C, e a última entre B e C, como descrito pela figura 4 a seguir.

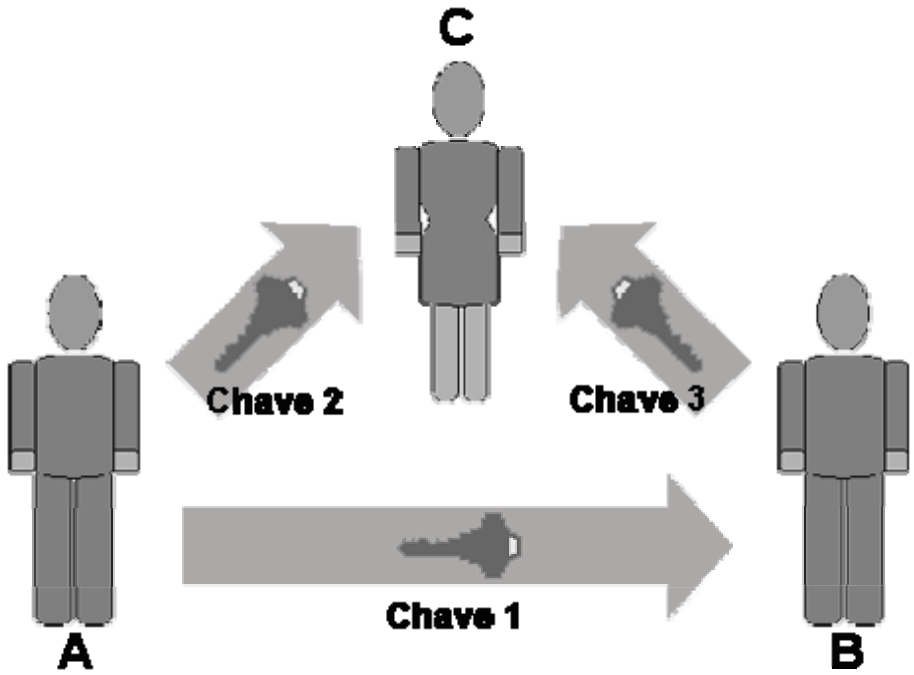

FIGURA 4 – Ilustração de um processo de criptografia por chaves secretas
Fonte: GARFINKEL, Simson; SPAFFORD, Gene (1999, p.199).

Se mais pessoas fossem incluídas neste sistema de comunicação, mais chaves seriam necessárias. No caso de mais duas pessoas, mais sete chaves seriam necessárias. Generalizando, quanto mais pessoas quisessem se comunicar utilizando chave secreta simétrica, um número muito grande de chaves seria necessário, pois a função que explicita o número de chaves é exponencial, gerando um grande problema para o gerenciamento de chaves utilizadas por grandes grupos de usuários.

Por sua vez, nos sistemas modernos de criptografia, as chaves simétricas são bem mais rápidas que os algoritmos de chave pública (relatados a seguir com mais detalhes). Além disso, as chaves simétricas podem ser implementadas mais facilmente. (GARFINKEL E SPAFFORD, 1999).

2.1.2.1 Principais Algoritmos Simétricos

- **DES**

O DES, *Data Encryption Standard* (Padrão de criptografia Digital), é um algoritmo padrão de criptografia há mais de 20(vinte) anos, sendo muito utilizado em aplicações bancárias até os dias atuais. Apesar de já apresentar alguns sinais de fraqueza, essencialmente, devido ao pequeno tamanho da chave, resistiu admiravelmente a anos de criptoanálise intensiva. Baseado em um algoritmo desenvolvido pela IBM, chamado Lúcifer, seu propósito foi criar um método padrão para proteção de dados.

Foi adotado pelo governo dos Estados Unidos em 1977, e como padrão do ANSI – *American National Standards Institute* (Instituto Nacional Americano de Normas). Este algoritmo é o mais amplamente usado internacionalmente ainda hoje, e foi um avanço científico significativo no sentido de ter se constituído no primeiro algoritmo de criptografia tornado público, pois até então todos os algoritmos do gênero eram secretos. (TERADA, 2000).

Fundamentalmente, o DES realiza somente duas operações sobre sua entrada: deslocamento e substituição de bits. A chave controla exatamente como esse processo ocorre. Ao fazer estas operações repetidas vezes e de uma maneira não-linear, chega-se a um resultado que não pode ser revertido à entrada original sem o uso da chave.

O algoritmo possui 64 bits, tanto de entrada como de saída e uma chave de 56 bits, sendo armazenada em 8 bytes para incluir bits de paridade.

Ao longo da década de 1980, o consenso dos criptógrafos era de que o DES não tinha nenhuma fraqueza: utilizando “ataque de força bruta” – pois, para quebrar uma chave de 56 bits, um número entre 0 e 73 quatrilhões aproximadamente, de combinações possíveis, mesmo o computador mais rápido da época, levaria anos para decifrar uma única mensagem, este fato deixava transparecer que o DES não deixava margem à quebra de sigilo.

Mas, a partir da década de 1990, os computadores se tornaram mais rápidos e, conseqüentemente, mais ágeis para montar em um curto período o texto

cifrado, o que mostrou que o DES não seria assim “tão forte”. Em 1999, na RSA *Conference*, a *Eletronic Frontier Foundation* quebrou uma chave de DES em menos de 24 horas. (BURNETT E PAINE, 2002).

Como na informática o avanço na velocidade de processamento dos computadores é uma constante, logo este tempo pôde ser reduzido em minutos, deixando bem claro que o DES precisava de um substituto.

- **Triple DES**

Um substituto ao DES, amplamente utilizado é o chamado “Triple DES”. Como sou próprio nome já diz, seu diferencial em relação ao DES é a utilização do algoritmo de criptografia por três vezes, com chaves diferentes de 56 bits em cada etapa. Essencialmente, isso seria a mesma coisa que utilizar uma chave de 168 bits. Pode se pensar que, para quebrar uma chave no DES, demoraria um tempo de 24 horas, enquanto que no Triple DES demoraria 72 horas. Isso estaria correto se o invasor soubesse que a primeira chave foi quebrada, mas isto só acontecerá se este tiver descoberto a primeira chave até que a combine com as outras duas corretas, prolongando muito o tempo para quebrar as três chaves pelo “ataque de força bruta”.

Segundo Burnett e Paine (2002 p. 40), o Triple DES apresenta dois problemas: o primeiro é que os analistas descobriram uma maneira de simplificar o “ataque de força bruta”, reduzindo-o de maneira inteligente, equivalendo a um ataque de força bruta a uma chave de apenas 108 bits ao invés de 168 bits. Apesar de uma chave de 108 bits ser considerada segura, porém para alguns estudiosos, esta “fraqueza” incomodava.

O segundo problema é a velocidade, pois a DES leva muito tempo para criptografar e decriptografar os dados e o Triple DES levaria três vezes mais tempo para realizar estas tarefas. Novamente torna-se evidente a necessidade de um novo algoritmo.

- **AES**

O AES *Advanced Encryption Standard* (Padrão de criptografia Avançada) foi desenvolvido para substituir o DES. No dia 2 de Janeiro de 1997, o órgão oficial

norte-americano NIST – *National Institute of Standards in Technology* (Instituto Nacional de Normas em Tecnologia) anunciou formalmente um plano para definir um algoritmo como o novo padrão para criptografia, convidando qualquer pessoa a desenvolver um algoritmo com um novo padrão que seria conhecido como AES, abrindo assim uma espécie de “concurso”, com a condição que o vencedor não teria quaisquer direitos quanto à propriedade intelectual do algoritmo selecionado. Os critérios para avaliação dos algoritmos que viessem a concorrer seriam: segurança (sem nenhuma fraqueza algorítmica), desempenho (rápido em várias plataformas) e tamanho (não poderia ocupar muito espaço nem utilizar muita memória). (BURNETT E PAINE, 2002).

Várias pessoas físicas e jurídicas desenvolveram seus algoritmos, e no dia 20 de agosto de 1998, o NIST selecionou 15 candidatos. Vários dos 15 algoritmos originais não duraram muito tempo, pois em alguns foram descobertas fraquezas e em outros simplesmente eram muito grandes ou muito lentos, reduzindo a lista em apenas 5 algoritmos até agosto de 1999, sendo eles o MARS, RC6, Rijndael, Serpent e Twofish. No ano seguinte, estes algoritmos foram testados para decidir qual deles era o vencedor. Finalmente, no dia 2 de outubro de 2000, o NIST anunciou como o grande vencedor o algoritmo Rijndael, desenvolvido por 2 pesquisadores belgas, Vincent Rijmen e Joan Daemen (BURNETT E PAINE, 2002).

O AES usa um número variável de tamanho de chave e tamanho de bloco. Atualmente, sua especificação de trabalho é possuir chaves com tamanhos que variam entre 128, 192 e 256 bits, criptografando blocos de 128, 192 e 256 bits (todas as nove combinações de tamanho de chave e tamanho de blocos são possíveis). Além disso, esses números podem ser facilmente expandidos para múltiplos de 32 bits. O AES possui facilidade de implementação, propiciando uso em *Smart Cards* e outros equipamentos que utilizam pouca memória RAM; além disso, utiliza poucos ciclos de processamento.

O código desse algoritmo é bem enxuto e não depende de nenhum outro tipo de componente criptográfico, como números randômicos. Esse aspecto faz com que sua utilização apresente um nível de segurança superior. (STOHLER, 2002).

- **RC2, RC4, RC5 e RC6**

Estes algoritmos foram desenvolvidos pelo Professor Ronald Rivest em 1987 para a empresa RSA Data Security.

Inicialmente estes algoritmos foram mantidos secretos através de acordos de sigilo, porém o RC2 foi revelado por uma mensagem anônima na Usenet em 1994, assim como o RC4 em 1996. Ambos são implementações que permitem a utilização de chaves de 1 a 2048 bits, embora, muitas vezes, o tamanho da chave seja limitado a 40 bits no *software* vendido para exportação.

Com chaves pequenas (menores que 48 bits), tanto o RC2 quanto o RC4 são códigos fáceis de serem quebrados, e não se tem muita informação sobre sua segurança com chaves extensas. O RC2 é uma cifra de bloco, similar ao DES. O RC4 é uma cifra de corrente, onde o algoritmo produz uma corrente de pseudonúmeros que são cifrados através de uma operação lógica XOR, junto com a própria mensagem. O RC5 permite que o tamanho da chave, o tamanho de blocos de dados e o número de vezes que a criptografia será realizada sejam definidos pelo usuário. (GARFINKEL E SPAFFORD,1999).

Proveniente do RC5, o RC6 possui uma chave de 8 a 1024 bits, sendo um algoritmo simples o suficiente para ser memorizado e podendo ser facilmente implementado de forma compacta, tanto em *software* quanto em *hardware*, no “concurso” estabelecido pelo NIST para o desenvolvimento do AES. Este algoritmo ficou entre os 5 finalistas selecionados, conforme citado anteriormente.

- **IDEA**

O IDEA – *International Data Encryption Algorithm* (Algoritmo de Criptografia de Dados Internacionais) é um algoritmo de cifragem de bloco desenvolvido em Zurique, na Suíça, por James L. Massey e Xuenjia Lai, e tornado conhecido através de publicação, em 1990 (GARFINKEL E SPAFFORD, 1999). É um algoritmo que utiliza uma chave de 128 bits e tanto o texto legível (entrada) como o texto ilegível (saída) utilizam 64 bits. Foi projetado para ser eficiente em implementações por *software* e possui patente nos EUA e na Europa da Ascom-Tecvh AG.

O IDEA possui uma estrutura semelhante ao DES, com um número fixo de iterações de uma mesma função, utilizando sub-chaves distintas e possuindo o

mesmo algoritmo com a finalidade de criptografar e decriptografar, alterando-se, em relação àquele, na forma de geração de sub-chaves. (STALLINGS, 1999). Portanto, não foi desenvolvido com o objetivo de substituir o DES. Como seus direitos são patenteados, é necessário de uma licença para ser utilizado.

É um algoritmo que ainda não pode ser conceituado como forte, devido a seu pouco tempo de vida, porém aparenta ser robusto. Sua chave de 128 bits elimina a possibilidade de um invasor utilizar computadores atualizados mais velozes para efetivar-lhe “ataques por força bruta”. (TERADA, 2000).

2.1.3 Criptografia Assimétrica

A criptografia de chave simétrica até pode manter seguros os dados contra invasores, porém quando se necessita compartilhar as informações criptografadas com outras pessoas, é inevitável o compartilhamento da chave usada na criptografia para permitir decriptografar os dados posteriormente, tornado-os legíveis novamente. Isso pode gerar um certo constrangimento em relação à segurança dos dados criptografados, pois como foi visto anteriormente, na criptografia de chave simétrica, a mesma chave utilizada para criptografar os dados também é utilizada para decriptografá-los. Este procedimento gera um problema quanto à distribuição da chave.

Já na criptografia assimétrica, também conhecida como criptografia de chave pública, isso não ocorre, pois os problemas de distribuição de chaves existentes na criptografia por chave simétrica são sanados com a utilização da chave pública (que será melhor detalhada a seguir), não havendo a necessidade do compartilhamento de uma mesma chave nem de um pré-acordo entre as partes interessadas. Com isto o nível de segurança é maior. (SCHNEIER, 1996).

A criptografia assimétrica foi desenvolvida inicialmente em meados da década de 70 pelos pesquisadores Whitfield Diffie e Martin Hellman, da universidade de Stanford, abordando uma nova maneira de enviar chaves seguramente. Este conceito utiliza duas chaves diferentes, embora relacionadas entre si. O relacionamento é matemático; o que uma chave criptografa a outra decriptografa.

Ambas as chaves são necessárias para bloquear e desbloquear os dados, podendo uma delas se tornar pública sem colocar a segurança em perigo. Esta chave é conhecida como *Chave Pública*, sua contraparte é chamada de *chave privada*. Para criptografar os dados é utilizada a chave pública, e para decriptografá-los é utilizada a chave privada. (BURNETT E PAINE, 2002).

A figura 5, a seguir, demonstra um processo que cifra uma mensagem, utilizando a chave pública, e a decifra utilizando a chave secreta(pública) relacionada.

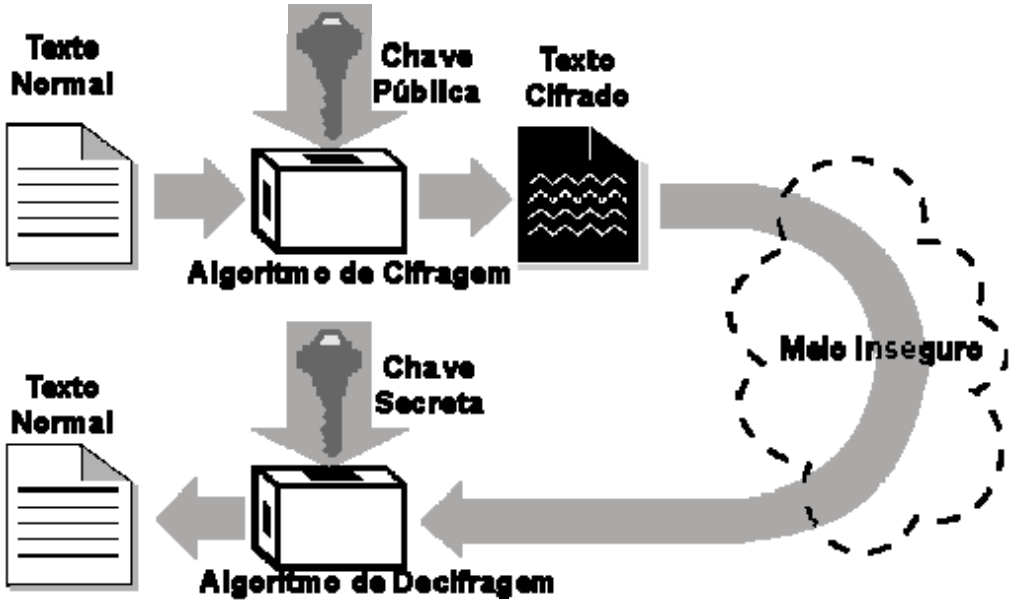

FIGURA 5 – Ilustração de um processo de criptografia por chave pública
Fonte: GARFINKEL, Simson; SPAFFORD, Gene (2001, p.208).

Utilizando a criptografia de chave assimétrica, o invasor ou interceptador da mensagem criptografada, mesmo tendo em mãos a chave pública, não consegue decifrar a mensagem, pois a chave utilizada para cifrar não é a mesma utilizada para decifrar a mensagem.

A figura 6, demonstra um processo de criptografia, utilizando-se chave pública e privada no envio e recebimento da mensagem.

A mensagem cifrada pelo emissor, utilizando a chave privada, só poderá ser decifrada pelo receptor utilizando a sua chave pública relacionada. Caso o receptor venha a enviar uma mensagem cifrada ao emissor, utilizando a sua chave pública relacionada, essa mensagem só poderá ser decifrada utilizando-se a chave privada do próprio emissor.

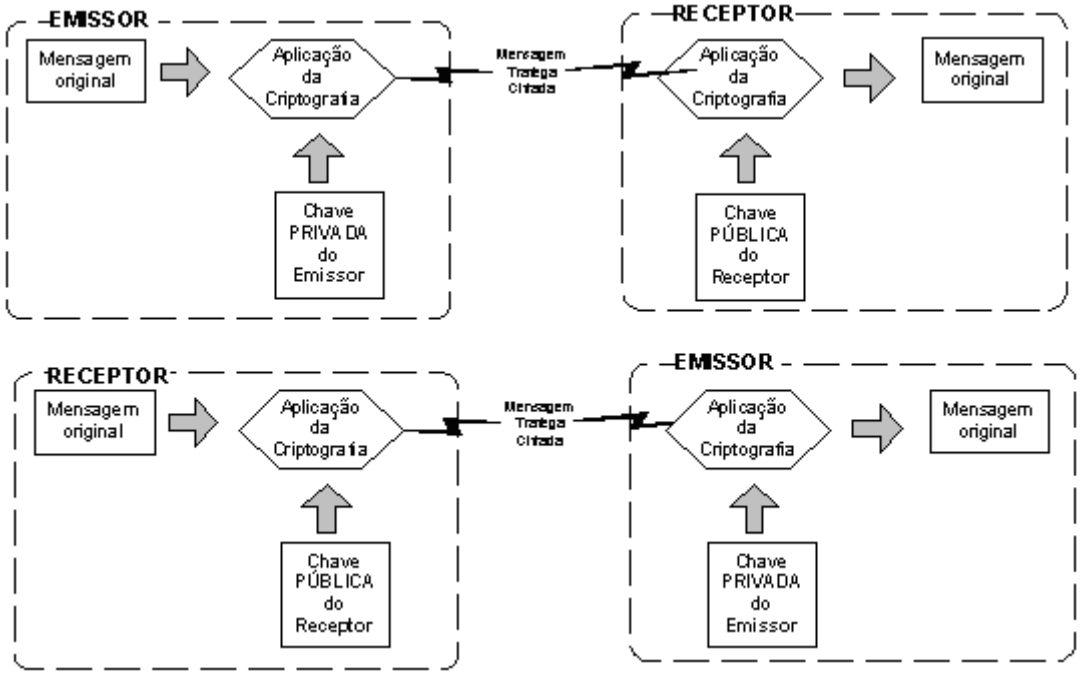

FIGURA 6 – Ilustração de um processo de criptografia por chave pública
Fonte: VOLPI, Marlon M. (2001 p.14)

Caso a mensagem enviada pelo emissor e sua respectiva chave pública forem interceptadas, o interceptador não terá condições de criar uma nova mensagem para o receptor, levando em consideração que a codificação através da chave pública somente permite a leitura do possuidor da chave privada, o que não é o caso do receptor. No entanto, o interceptador poderia enviar uma mensagem para o emissor, utilizando a chave pública deste, o qual possui a chave privada, permitindo a leitura da mensagem. Para que esse processo fosse evitado, seria necessário que tanto o emissor quanto o receptor tivessem, cada um deles, uma chave privada e uma pública.

Segundo Garfinkel e Spafford (2001), além do ganho da segurança em relação às chaves secretas, existe uma conveniência adicional na chave pública, uma vez que ela nunca precisa ser repetida ou revelada. Outra vantagem é a sua capacidade de prover assinaturas digitais, para comprovar a origem e a integridade dos dados. No entanto, as chaves públicas apresentam um problema significativo: são muito lentas. Na prática, os algoritmos criptográficos de chave pública (assimétrico) são entre 10 e 100 vezes mais lentos do que os equivalentes de chave secreta (simétrico).

No quadro 1 a seguir, é ilustrado o resumo de alguns dos aspectos importantes da criptografia convencional (simétrica) e de chave pública (assimétrica).

Quadro 1 – Principais diferenças entre Encriptar Convencionalmente ou com Chave Pública

Encriptar Convencionalmente	Encriptar com Chave Pública
Necessidade para trabalhar:	
O mesmo algoritmo e a mesma chave são usados para a criptografar e decriptografar.	Um algoritmo é usado para criptografar e decriptografar com um par de chaves: uma para criptografar e a outra para a decriptografar.
O emissor e o receptor devem compartilhar o algoritmo e a chave.	O emissor e o receptor devem ter o par de chaves (privada e pública).
Necessidade para a Segurança:	
A chave deve ser mantida em segredo.	Uma das duas chaves deve ser mantida em segredo.
Deve ser impossível ou pelo menos impraticável decifrar uma mensagem se nenhuma outra informação estiver disponível.	Deve ser impossível ou pelo menos impraticável decifrar uma mensagem se nenhuma outra informação esteja disponível.
Conhecimento do algoritmo mais as amostras do texto cifrado devem ser insuficientes para determinar a chave.	Conhecimento do algoritmo mais uma das chaves, mais as amostras do texto cifrado devem ser insuficientes para determinar a outra chave.

Fonte: Stallings, Willian (1999, p.167)

2.1.3.1 A Segurança nos Algoritmos Assimétricos

Comparando novamente o algoritmo Simétrico com o Assimétrico, no primeiro, a maneira mais rápida para violar a segurança seria através de “ataques por força bruta”, conforme já mencionado anteriormente. Já no segundo existem outros métodos mais rápidos de violar a segurança e obter o resultado desejado pelo invasor. Analisando de modo geral, poderia chegar a conclusão que o algoritmo Simétrico é mais seguro que o Assimétrico. No entanto, em um “ataque por força bruta”, o invasor pode conseguir a violação do algoritmo Simétrico logo na primeira tentativa, porém Burnett e Paine (2002, p. 78), relatam que ninguém foi capaz de desenvolver um algoritmo Assimétrico que não tenha fraquezas, sendo que para todos os algoritmos de chave pública, existem técnicas (atalhos) que quebrarão a segurança mais rapidamente do que um “ataque por força bruta”. Contudo, existem duas razões que a maioria dos usuários estão dispostos para conviver com estes

“atalhos”. A primeira diz que os criptógrafos realizaram uma grande pesquisa para quantificar o tempo requerido pelos atalhos e concluíram que, mesmo um algoritmo sendo suscetível a um ataque mais rápido que o de “força bruta”, este tempo ainda é muito longo, sendo suficiente para a maioria dos usuários quando se trata de segurança. A segunda razão que faz os usuários utilizarem o Algoritmo Assimétrico suscetíveis a atalhos diz que estes algoritmos são a melhor maneira de resolver o problema de distribuição de chaves.

2.1.3.2 Principais Algoritmos Assimétricos

A seguir serão abordados os três algoritmos amplamente utilizados para resolver o problema de distribuição de chaves, sendo eles o DH, RSA e ECDH.

- **DH**

O algoritmo DH (Diffie-Hellman) foi desenvolvido pelos mesmos pesquisadores que publicaram o artigo referente à criptografia de chave pública, Whitfield Diffie e Martin Hellman, abrindo caminho para os algoritmos Assimétricos.

O algoritmo DH não gera uma chave de sessão simétrica e a distribui utilizando a tecnologia de chave pública; em vez disso, a tecnologia de chave pública é utilizada para gerar a chave de sessão simétrica, possibilitando que cada parte correspondente possua um valor secreto e um valor público. Combinado um valor privado com outro público, cada parte gerará o mesmo valor secreto, como ilustrado na figura 7. (BURNETT E PAINE, 2002).

Segundo Burnett e Paine (2002), o algoritmo DH não criptografa os dados, ele somente gera um segredo. Dessa forma, as duas partes podem gerar o mesmo segredo e então utilizá-lo para criar uma chave de sessão para ser utilizada em um algoritmo simétrico. Este procedimento é chamado de *acordo de chaves*, onde as duas partes estão de acordo sobre qual chave deve ser utilizada.

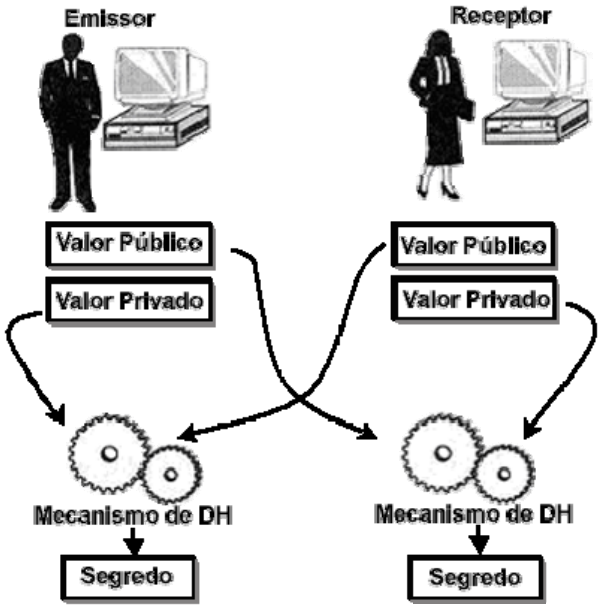

FIGURA 7 – Ilustração do algoritmo DH para gerar o mesmo valor secreto
FONTE: BURNETT, Steve; PAINE, Stephe. (2002, p.90).

A técnica DH é baseada no Problema do Logaritmo Discreto. Uma chave Diffie-Hellman consiste em um gerador, um módulo e um valor público, onde a chave privada é o mesmo módulo junto com o valor privado. Este algoritmo utiliza apenas um número primo de 1.024 bits como módulo. Este é um fator que o faz tão seguro e complexo, comparando com o algoritmo RSA que veremos a seguir.

- **RSA**

O algoritmo RSA foi tornado público em 1978 por Ronald Rivest, Adi Shamir e Len Adleman, que na época trabalhavam no *Massachussets Institute of Technology* (M.I.T.). As iniciais RSA correspondem às iniciais dos inventores do código. (COUTINHO, 2000).

O algoritmo tomou por base o estudo feito por Diffie e Hellman, porém usando outro fundamento matemático para a criação das chaves públicas. Ao contrário do algoritmo DH, que utiliza um único número primo de 1.024 bits como módulo e realiza o acordo de chave, o módulo de 1.024 bits do RSA é obtido através da multiplicação de dois números primos de 512 bits e criptografa os dados.

Segundo Terada (2000 p.101), o algoritmo RSA é baseado na dificuldade computacional de fatorar um número inteiro em números primos. Desta forma, a promessa alcançada pelo algoritmo RSA é a facilidade de multiplicar dois números

primos para obter um terceiro número, mas muito difícil recuperar os dois primos a partir daquele terceiro número.

Para implementar o RSA são necessários dois parâmetros básicos: dois números primos (*p* e *q*) e seu produto (*n*). A chave de codificação do RSA é constituída essencialmente pelo número *n*, sendo $n = p \cdot q$. Cada usuário do método possui sua chave de codificação, que é tornada pública (*n*), por isso *n* é conhecido como chave pública e a chave de decodificação é constituída pelos primos *p* e *q* (chave privada), que o usuário deve manter em sigilo, pois, caso contrário a segurança do método estará comprometida.

Se a fórmula $n = p \cdot q$ é conhecida, a princípio parece ser fácil descobrir *p* e *q*, a partir do número conhecido *n*, pois bastaria fatorar *n* para que o código fosse decifrado. No entanto, para se obter chaves de codificação seguras é necessário que estas sejam geradas por números muito grandes (com 150 algarismos ou mais).

Segundo Coutinho (2000, p.3), desde sua criação, o algoritmo RSA tem reinado como a única abordagem amplamente aceitável e implementável a criptografia de chave pública, utilizando os princípios de uma expressão exponencial, aliados a conceitos dos números primos, aritmética modular e do Teorema de Euler.

O esquema de criptografia do algoritmo RSA é aplicado sobre blocos de tamanho máximo definido. Como a maioria das mensagens supera este tamanho, elas devem ser quebradas em fragmentos de tamanho apropriado.

É importante notar que, quanto mais avançadas forem as técnicas de fatoração, torna-se razoável admitir que não haverá modificação na utilidade destes algoritmos, pois os números primos são gerados pela utilização das mesmas técnicas. Logo, quanto mais fácil for fatorar um número de um tamanho específico, mais fácil será gerar primos de tamanhos maiores.

Considerando que a busca de fatores primos é um dos problemas mais antigos da matemática, sua utilização para um problema prático é considerada de grande elegância. A matemática que estuda as propriedades dos números inteiros é conhecida como Teoria dos Números (COUTINHO, 2000).

- **ECDH**

O algoritmo ECDH (*Elliptic Curve Diffie-Hellman*), é o resultado de uma curva elíptica *Elliptic Curve (*EC). Apesar da curva não ser a única forma que uma EC possa ter, ela é a mais comum. Analisando a fundo esta forma, nem mesmo é uma EC criptográfica, mais quando os criptógrafos falam sobre uma EC, eles geralmente mostram uma figura semelhante à figura 8.

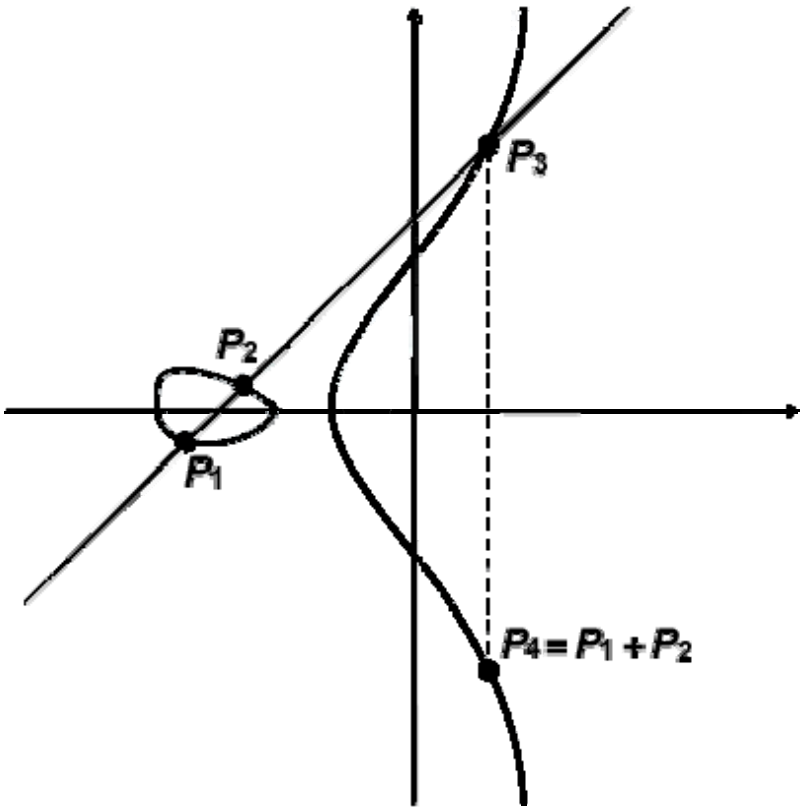

FIGURA 8 – Ilustração de uma Curva Elíptica e a adição de uma EC.
FONTE: BURNETT, Steve; PAINE, Stephe. (2002, p.94).

Burnett e Paine (2002), relatam que os criptógrafos utilizam apenas alguma das diversas versões de Ecs, sendo que as curvas utilizadas por eles caem em duas categorias principais, geralmente denominadas de “ímpar” e “par”. Uma EC criptográfica é considerada discreta, ou seja, possui apenas inteiros, não existindo ponto fracionário ou decimal. Todos os números caem em um certo intervalo, sendo que quanto maior for o intervalo, mais segura é a curva e, quanto menor for o intervalo, mais rápido serão as computações.

Como pode se observar na figura 8, uma curva elíptica possui pontos, sendo que um ponto é uma coordenada de x, y. O ponto rotulado como P_3 também poderia ser descrito como (3,8), sendo que a coordenada x = 3 e y = 8. A figura 8

também ilustra que é possível adicionar pontos em uma EC. Localizando dois pontos que se deseje adicionar, desenha-se uma linha utilizando estes pontos e examina-se onde essa linha secciona a EC.

A forma gráfica para as curvas elípticas pode ser descrita como equações matemáticas, trabalhando com números e equações ao invés de figuras, permitindo, assim, escrever programas de computadores que realizam o trabalho, e se estes programas manipularem números, talvez obtenha a criptografia, necessitando de uma função de via única chamada de “multiplicação escalar”. (BURNETT E PAINE, 2002).

Na prática, quando se deseja enviar uma mensagem utilizando o algoritmo ECDH, o remetente da mensagem pega a chave pública do destinatário contendo informações suficientes para gerar seu próprio par de chaves temporários de ECDH. Agora tanto destinatário quanto remetente possui um par de chaves relacionada com as chaves pública e privada. Desta forma o remetente utiliza a sua chave privada e a chave pública do destinatário para gerar um “ponto secreto na curva”, utilizando de alguma forma este ponto secreto como uma chave de sessão. Sabendo que um ponto é uma coordenada (x,y), os dois correspondentes terão que decidir antecipadamente quais bits, a partir deste número, devem ser utilizados como chave. A figura 9 ilustra a combinação de chaves entre o remetente e o destinatário da mensagem para obter um ponto secreto.

Para ler a mensagem, o remetente necessita da chave de sessão, que é obtida através da combinação da sua chave privada com a chave pública temporária do remetente, enviada junto com a mensagem criptografada. Caso um invasor tenha acesso à mensagem criptografada, o mesmo teria que possuir uma das duas chaves privadas, pois o fato de conhecer as duas chaves públicas não resolve a questão.

O algoritmo ECDH se parece com o algoritmo DH, pois ambos combinam chave pública e privada de maneira especial para gerar um segredo compartilhado, a principal diferença está na matemática subjacente, e isso explica o nome Elliptic Curve Diffie-Hellman (BURNETT E PAINE, 2002).

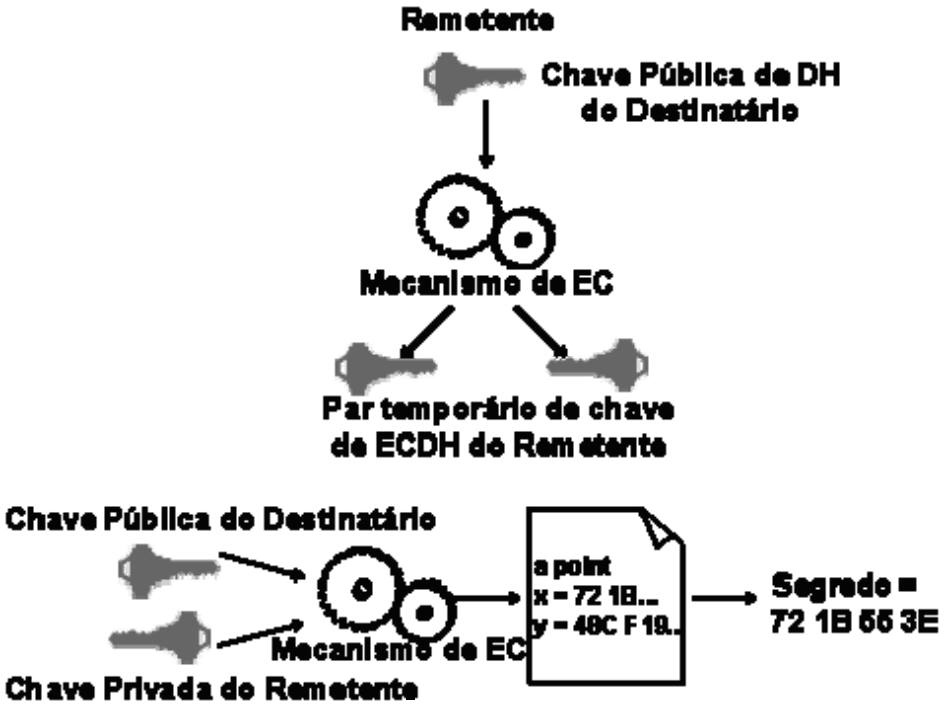

FIGURA 9 – Ilustração da e uma combinação de chaves entre remetente e destinatário de uma mensagem para obter um ponto secreto.
FONTE: BURNETT, Steve; PAINE, Stephe. (2002, p.98).

2.2 Assinatura digital

Como citado anteriormente no capítulo 2.1.3 desta dissertação, a criptografia de chave pública ajuda a resolver o problema de distribuição de chaves, mais também pode ajudar a resolver o problema de autenticação e de não-repúdio de mensagens e documentos transitados por meios eletrônicos. A assinatura digital é uma maneira de implementar estes recursos e seu conceito é aplicado para documentos digitais, comparando com a maneira de assinatura comum para documentos impressos; ambas autenticam a origem dos dados contidos nos documentos.

A assinatura escrita pode ser explicada como uma única combinação de traços de lápis ou caneta que seja bem difíceis para outra pessoa qualquer forjar, e com o passar do tempo, torna-se parte da identidade do indivíduo. (VOLPI, 2001).

A assinatura digital pode ser entendida como uma identificação composta por números, que podem ser empregados como um meio efetivo na proteção de

informações, estejam elas armazenadas em um computador ou transmitidas pela rede. No segundo caso, a assinatura digital busca garantir que determinada mensagem não seja alterada durante seu trajeto.

Para esclarecer um processo de assinatura digital, é melhor recordar como é feito o processo de criptografia, utilizando um algoritmo RSA. Quando se utiliza este algoritmo com intuito de criptografar um documento, tornando-o seguro contra leitura e entendimento de terceiros. É imprescindível que utilize uma chave pública para criptografar este documento, que apenas pode ser decriptografado utilizando a chave privada equivalente.

Na assinatura digital, este processo é feito de modo inverso, pois o algoritmo RSA permite o funcionamento tanto do privado ao público, como do público ao privado. Sendo assim, os dados são criptografados utilizando a chave privada, e apenas a chave pública pode ser utilizada para decriptografá-los. Esta técnica não serve como garantia de segurança contra leitura dos dados criptografados, pois sendo utilizada uma chave pública para decriptografar, qualquer pessoa com a posse desta chave poderá ter acesso aos dados de maneira legível. Porém, esta técnica serve para assegurar o conteúdo de um documento, pois se uma chave pública decriptografar um documento adequadamente, isso significa que este documento só pode ter sido criptografado com a chave privada equivalente. Desta forma qualquer documento digital, sendo ele um texto, uma mensagem, uma gravura, quando criptografado utilizando uma chave privada, esta técnica é convencionalmente chamada pela comunidade de criptografia como “assinatura digital”. (BURNETT E PAINE, 2002).

Segundo Volpi (2001), a assinatura digital é efetivada através de algoritmos de autenticação, através de um processo lógico-matemático aplicado sobre a mensagem a ser transmitida, criando, assim, uma determinada expressão que será utilizada como assinatura.

Conforme ilustrado na figura 10 abaixo, a mensagem original é acompanhada de uma assinatura digital, baseada na chave privada do remetente. Quando a mensagem chega a seu destino, a assinatura é verificada utilizando-se a chave pública que pertence ao remetente. Confirmada a assinatura digital a partir da verificação, pode-se ter a certeza da autenticidade da mensagem e do remetente.

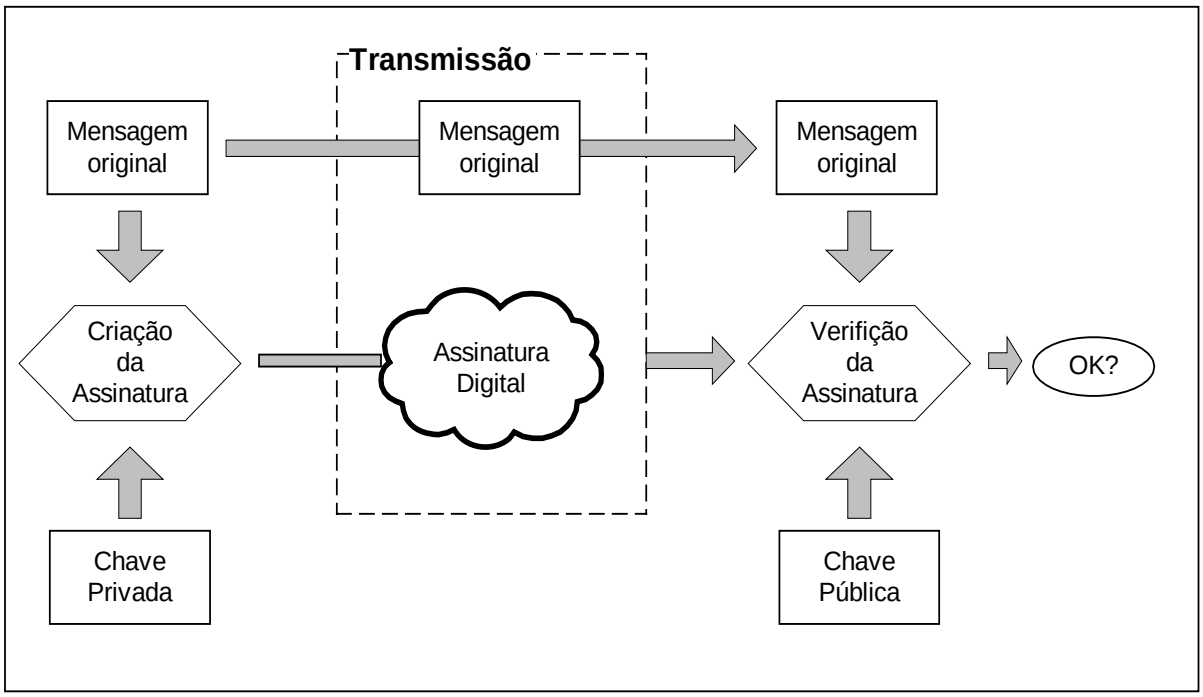

FIGURA 10 – Ilustração do funcionamento da Assinatura Digital.
FONTE: VOLPI, Marlon M. (2001, p.17).

Segundo Burnett e Paine (2002), as assinaturas digitais dependem de três suposições fundamentais, sendo elas:

1) A chave privada deve ser segura
2) Apenas o proprietário da chave privada tem acesso a ela (não existe nenhuma suposição técnica, exceto que as chaves devem ser protegidas);
3) A única maneira de produzir uma assinatura digital seja utilizando a chave privada (examinando por um ponto de vista da matemática, pode-se demonstrar que uma assinatura é única).

A figura 11, ilustra a unicidade de uma assinatura digital, dando três exemplos A, B e C, onde (A) mostra o caminho que os dados tomam para se tornarem uma assinatura digital e para serem verificados, (B) mostra um possível ataque de um invasor tentando iniciar com um texto simples, encriptá-lo com uma chave que não seja a chave privada verdadeira e ainda produzir um texto cifrado corretamente, ou talvez como no exemplo (C), o invasor tente realizar uma outra operação no texto simples, sem utilizar a criptografia normal RSA, possivelmente utilizando a chave pública como uma guia e ainda assim produzir um texto cifrado corretamente. Caso estes dois últimos exemplos fossem possíveis, uma assinatura digital não seria única. Não sendo única, não poderia afirmar que o proprietário da

chave privada estivesse confirmando o texto simples. Porém, os criptógrafos informam que ainda não se tem conhecimento de tal ataque bem sucedido, contendo na literatura frase como “computacionalmente inviável”. Apesar de ainda não ter provado completamente a unicidade de uma assinatura digital, pesquisadores já gastaram inúmeras horas tentando violar este item, e não chegaram nem mesmo perto de realizá-lo. (BURNETT E PAINE, 2002).

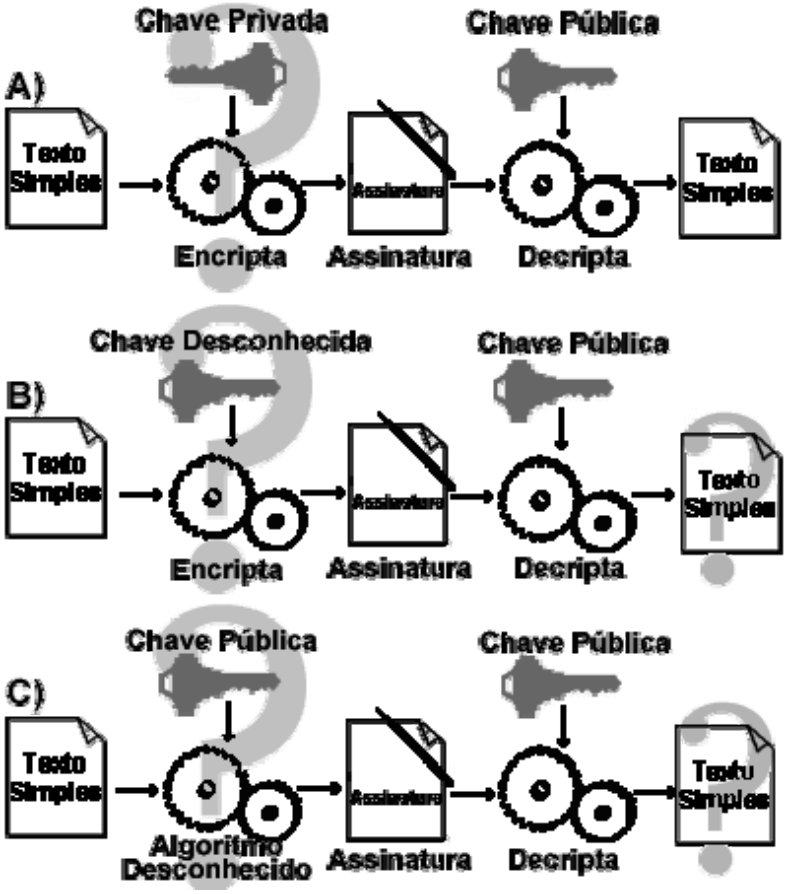

FIGURA 11 – Ilustração da unicidade da Assinatura Digital.
FONTE: BURNETT, Steve; PAINE, Stephe. (2002, p.119).

2.2.1 Resumos de mensagem

Como a assinatura digital é obtida através do uso da criptografia assimétrica ou de chave pública, ela sofre os mesmos problemas que uma criptografia de chave pública, ou seja, é muito mais lenta que uma criptografia de chave simétrica. Dependendo do tamanho do documento a ser assinado, pode demorar alguns

minutos ou horas para serem cifradas com uma chave privada. Encriptar um documento muito grande não é muito eficaz, pois reduziria o desempenho, portanto, em vez de encriptar um documento simples inteiro com a chave privada, o melhor método seria encriptá-lo com um “representante de dados”.

Na criptografia, o representante de dados é um “resumo de mensagem”. A palavra “resumo” significa que para condensar, ou reduzir e ser suficientemente seguro, o arquivo que se queira enviar deve passar por um processo de condensação. Desta forma, é possível dizer que um resumo de mensagem é um algoritmo que recebe qualquer comprimento de entrada e mescla essa entrada para produzir uma saída pseudo-aleatória de largura fixa (vinte bytes), freqüentemente chamada de função *hash*. A função *hash* pode significar desordem ou confusão, e descreve eficientemente o resultado de uma *message digest*. (NEGROPONTE, 1995).

Como citado anteriormente, devido à lentidão dos algoritmos assimétricos – que, em geral, são cerca de mil vezes mais lentos que os simétricos – a utilização da função *Hashing* como componente na implementação de assinaturas digitais, assim como para produzir um representante maior de dados é tão necessária que, quase sempre, uma chave de resumo torna-se imprescindível. (SCHNEIER, 2001).

A função *Hashing* é uma espécie de impressão digital e serve, portanto, para garantir a integridade do conteúdo da mensagem que representa. Assim, após o valor *hash* de uma mensagem ter sido calculado através do emprego de uma função *Hashing*, qualquer modificação em seu conteúdo – mesmo em apenas um bit da mensagem – será detectada, pois um novo cálculo do valor *hash* sobre o conteúdo modificado resultará em um valor *hash* bastante distinto (DINIZ, 1999). A probabilidade de duas mensagens diferentes produzirem o mesmo bloco deve ser praticamente nula. Assim, a função *Hashing* oferece agilidade nas assinaturas digitais, além de integridade confiável.

Observando os resumos de mensagens na figura 12 abaixo, pode até parecer gerados de forma aleatória, portanto se resumir o mesmo documento utilizando o mesmo algoritmo, mesmo sendo em dois computadores diferentes, sempre será obtido o mesmo resultado. Sendo assim, a saída de um algoritmo de mensagem é pseudo-aleatória. Como pode se observar na figura 12 abaixo, duas mensagens parecidas não são iguais, portanto serão gerados dois resumos de

mensagens completamente diferentes, mesmo que a diferença seja apenas um único byte, o resultado gerado será dramaticamente diferente.

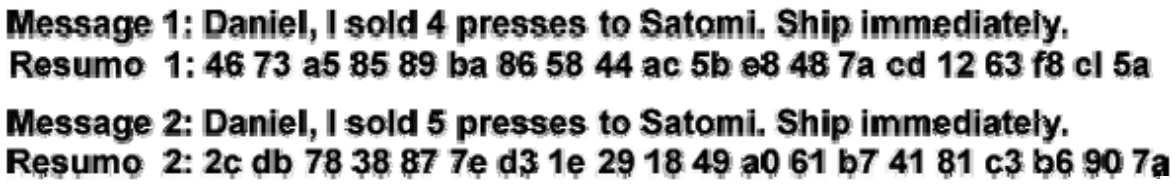

FIGURA 12 – Ilustração da unicidade da Assinatura Digital.
FONTE: BURNETT, Steve; PAINE, Stephe. (2002, p.120).

2.2.1.1 Principais algoritmos de resumo

Atualmente existem vários algoritmos de resumo, mas para Burnett e Paine (2002), três deles se destacam dos demais e dominam o mercado. Estes três algoritmos de resumo serão descritos a seguir, sendo eles o MD2, MD5 e SHA-1.

- **MD2**

Após Ron Rivest (um dos inventores do RSA) criar um algoritmo de resumo com o nome de MD, pensou que poderia melhorá-lo e desenvolveu a próxima geração, o MD2. Este algoritmo produz um resumo de 128 bits (16 Bytes), tendo 2^{128} possíveis valores de resumo. Este algoritmo foi amplamente utilizado, mas, com o passar dos anos, os analistas encontraram defeitos, como algumas colisões em certas classes de mensagens. Portanto o MD2 não tem sido muito utilizado, exceto em certificados antigos, não sendo mais indicado para a utilização em novos aplicativos.

- **MD5**

Após as descobertas dos analistas das fraquezas no MD2, Rivest começou a criar novos resumos. Foram criados por ele o MD3 que foi um fracasso, e quando

criou o MD4 logo se apresentou fraco, criando logo a seguir o MD5, sendo o melhor sucedido.

Garfinkel e Spafford (1999, p.203), relatam que: “assim como o MD2, o MD5 é um resumo de 16 bytes, no entanto o MD5 mostrou-se muito mais seguro e mais rápido do que os anteriores e tornou-se o algoritmo dominante e ainda é comumente utilizado”.

Embora largamente usado, foram descobertas algumas falhas nele em meados de 1996. O MD5 ainda não foi quebrado e ninguém encontrou colisões; em vez disso, algumas partes internas do algoritmo são vulneráveis. Se faltasse um ou dois componentes no algoritmo, ele poderia ser quebrado. (OAKS, 1999).

Pelo fato de algumas destas partes internas serem vulneráveis não significa que o algoritmo é fraco, pois ele funciona por inteiro e não por partes, porém alguns estudiosos dizem que não se quebra um algoritmo de uma só vez, quebra-o fragmento por fragmento. Sendo assim, faltam apenas alguns fragmentos para serem quebrados, e para evitar um colapso, seria melhor escolher um outro algoritmo.

- **SHA-1**

O SHA-1(*Secure Hash Algorithm*) foi desenvolvido com o intuito de melhorar algumas fraquezas encontradas na sua primeira versão, o SHA. Ambos foram desenvolvidos para utilização como padrão da assinatura digital pelo NIST – *National Institute for Standards Technology* (Instituto Nacional para Padrões e Tecnologia).

Apesar de sua semelhança com o MD5, o SHA-1 possui suas partes internas mais fortes, produzindo um resumo mais longo (160 bits comparados com 128 bits) e, por isso, é altamente recomendado pela comunidade criptográfica. Variantes do SHA-1 que produzem resumos de 192 bits a 256 bits ainda estão em desenvolvimento.

2.2.1.2 Utilização do algoritmo de resumo em uma assinatura digital

Segundo Diniz (1999), para se construir uma assinatura digital realmente confiável, ou seja, garantir sua integridade contra ataques, é necessário agrupar essa assinatura com um resumo e com um algoritmo.

Como a assinatura digital é obtida através do uso da criptografia de chave assimétrica ou chave pública, pode-se pensar que os algoritmos utilizados para realizar uma criptografia assimétrica, também podem ser utilizados para realizar uma assinatura digital. Em alguns casos, esta afirmação está correta, porém como a idéia de realizar uma assinatura digital utilizando estes algoritmos surgiu depois dos mesmos serem desenvolvidos, nem todos podem gerar uma assinatura digital.

O algoritmo DH, por exemplo, pode ser utilizado apenas para realizar a troca de chaves, e não de assinaturas digitais. Apesar do criptógrafo Taher El Gamal surgir com uma maneira de estender o DH para ser utilizado para encriptar e assinar, sua idéia nunca se tornou popular, previamente porque existia o algoritmo RSA que pode ser utilizado normalmente para criptografar quanto para assinar. Já no caso do algoritmo ECDH, utilizando a matemática de curva elíptica, ele pôde ser adaptado para criar assinaturas Existe inúmeras possibilidades de adaptação, porém a maneira mais comum de utilizar a EC para criar assinaturas é chamada de ECDSA. (BURNETT E PAINE, 2002).

2.2.2 Principais algoritmos utilizados em uma assinatura digital

Com o passar dos anos, vários algoritmos de assinatura foram propostos, porém segundo Burnett e Paine (2002), apenas o RSA, DSA e ECDSA se colocam entre os principais algoritmos em uso, e serão melhor detalhados a seguir.

- **RSA**

Este algoritmo já foi mostrado em detalhes anteriormente. O RSA é um algoritmo de chave pública que encripta um resumo com uma chave privada e produz uma assinatura digital. A matemática utilizada é a mesma tanto para a administração de chaves quanto para assinaturas digitais. Para forjar uma assinatura RSA é necessário conhecer a chave privada. Sem uma chave privada, ninguém foi capaz de produzir um fragmento de dados, chamar isso de uma assinatura digital e fazer com que isso seja verificado. (BURNETT E PAINE, 2002).

- **DSA**

O DSA (*Digital Signature Algorithm*) – Algoritmo de Assinatura Digital foi desenvolvido por David Kravitz quando este trabalhava para a NSA(*National Security Agency*) – Agencia de Segurança Nacional dos Estados Unidos – baseando-se no trabalho dos criptógrafos El Gamal e Claus Schnoor. Tornou-se o algoritmo oficial do governo dos Estados Unidos, e provavelmente seja o segundo mais usado hoje em dia, ficando atrás somente do RSA.

Segundo Volpi (2001, p.26), o DSA (*Digital Signature Algorithm*) é o algoritmo base para o chamado DSS (*Digital Signature Standard*). Após uma fase de implantação em que sofreu algumas poucas alterações, em 1994, veio a ser homologado pelo *Federal Information Processing Standard* (FIPS), tornando-se, assim, o método de assinatura digital padrão aceito pelo governo americano.

O DSA utiliza um resumo de dados, mas não o criptografa. Assim, sua utilização se restringe a assinaturas digitais.

O assinante resume a mensagem com o SHA-1 e trata desse resumo como um número relativamente grande de 160 bits de comprimento. Um outro número enviado ao algoritmo é um valor aleatório ou pseudo-aleatório, chamado de *k* e a última entrada é a chave privada. Na seqüência, o algoritmo realiza algumas operações matemáticas, uma das quais é a exponenciação modular. A saída são dois números, chamados de *r* e *s,* sendo esses dois números a assinatura. Utilizando o resumo como um número, junto com a chave pública e o *s*, o verificador realiza algumas operações matemáticas. O resultado dos cálculos é um número

chamado *v*. Se *v* for igual a *r*, a assinatura é verificada – considerada autêntica. Esse procedimento é mostrado na figura 13.

Duas partes do DSA produzem o mesmo número, utilizando uma entrada diferente, porém relacionados. Se algo for decomposto, as respostas finais irão diferir. Cada lado possui três entradas, o assinante tem o resumo, o *K* e a chave privada, o verificador tem o resumo, o *s* e a chave pública, sendo que os resumos têm que ser iguais e estarem relacionados, e a chave pública e a privada também têm que estar matematicamente relacionadas. O *k* e o *s*, apesar de não serem o mesmo número também estão relacionados Caso isso não ocorra, os dois agentes produzirão respostas finais diferentes, verificando a autenticidade da assinatura. (BURNETT E PAINE, 2002).

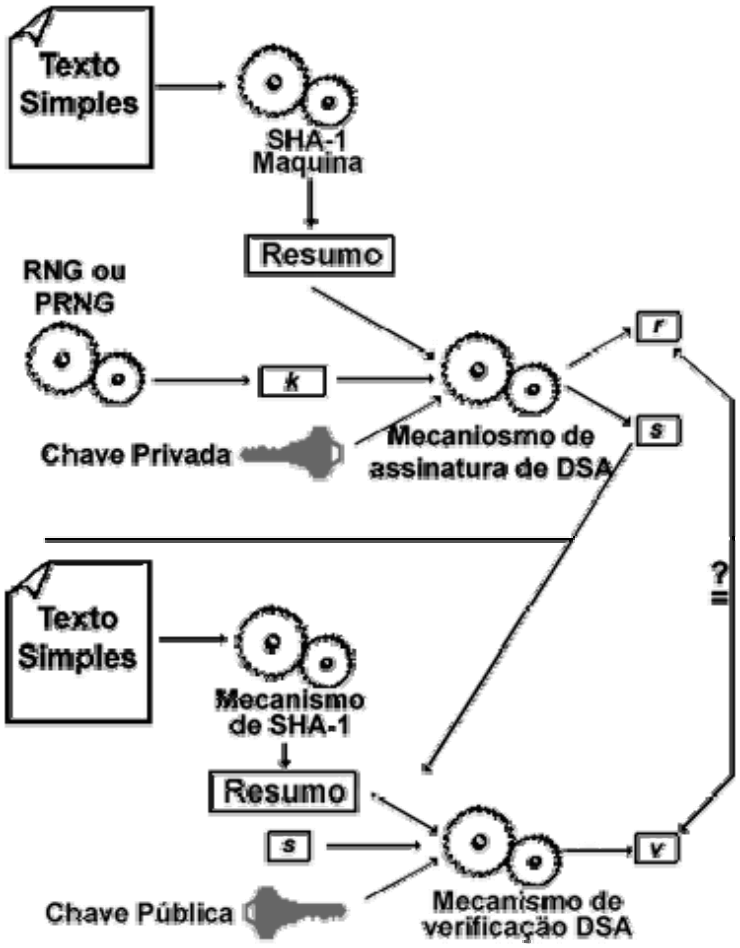

FIGURA 13 – Ilustração de um processo de criptografia utilizando o DSA.
FONTE: BURNETT, Steve; PAINE, Stephe. (2002, p.137).

A segurança do DSA reside no problema do logaritmo discreto, o mesmo problema que fornece a segurança ao DH. O tamanho das chaves geradas pelos algoritmos RSA e DSA é o mesmo (1.024 bits) (OAKS, 1999).

- **ECDSA**

O ECDSA (*Elliptic Curve Digital Signature Algorithm*) – Algoritmo de assinatura digital de curva elíptica – se parece bastante com o DSA . O assinante tem três entradas, a saída é *r* e *s,* e seu funcionamento é igual ao DSA, contendo até o mesmo tamanho da chave, porém a matemática subjacente do ECDSA são curvas elípticas. (BURNETT E PAINE, 2002).

2.3 Utilização da assinatura digital

Com o avanço da informática, e a disseminação das redes de computadores em todo mundo como a *internet*, por exemplo, possibilitou o uso de transações eletrônicas de diversos tipos, desde o envio de um simples e-mail até a compra de um automóvel, pode ser feito através destas transações, porém a grande questão é: como confiar na segurança destas transações? No mundo real do papel e da caneta, esta segurança é alcançada através de uma assinatura realizada pelo autor do documento, sendo esta assinatura única e de difícil falsificação, comprovando assim a autoria do documento assinado. Houve então a necessidade de trazer esta forma de autenticar e garantir a autoria de um documento para o mundo digital. O que antes era realizado utilizando papel e caneta, agora passou a ser realizado na forma digital, atribuindo a mesma confiança já consolidada no papel. (MARCACINI, 1999).

Através das técnicas que utilizam assinatura digital, foram alcançados alguns quesitos de segurança atendidos também pelos documentos tradicionais, como integridade e autenticidade. Por serem únicas, as assinaturas digitais permitem a validação da identidade do assinante e a autenticação do texto assinado, garantindo que o texto não seja alterado durante o trajeto eletrônico; e caso seja

alterado, a assinatura digital poderá ser automaticamente invalidada como visto anteriormente.

Referente ao avanço tecnológico e à quantidade crescente de transações realizadas por meio eletrônico, Burnett e Paine (2002, p.251), relatam que:

> Com o passar dos dias, as transações baseadas no papel - incluindo acordos de poder jurídico – estão se tornando obsoletas à medida que o uso de acordos eletrônicos transmitidos pela Internet cresce em popularidade. O principal motivo dessa alteração é a conveniência.

Normalmente, as assinaturas digitais utilizam os níveis mais elevados de codificação e são praticamente invisíveis para o usuário final que está utilizando as transações eletrônicas. Cada vez que uma mensagem é enviada ou recebida, ou um cartão de crédito é utilizado para uma transação no mundo digital, existem grandes chances de que haja alguma forma de tecnologia de assinatura digital validando e fornecendo segurança à transação (ALBERTIN, 1999).

As transações eletrônicas mais comuns de se utilizar a assinatura digital serão abordadas a seguir.

2.3.1 Comércio eletrônico

Nos dias atuais, muito se fala sobre a quantidade de negócios realizados no mundo virtual. É comum ler alguma matéria em revistas de tecnologia falando sobre a quantidade de transação de uma determinada empresa, utilizando alguma forma de comércio eletrônico.

O comércio eletrônico, também conhecido como *e-commerce*, engloba todas as atividades realizadas para vender produtos ou serviços através de algum meio eletrônico, porém foi através da *internet* que este tipo de transação ficou amplamente conhecido, sendo suas empresas denomidadas como “empresas pontocom". Um dos seus objetivos é atender, direta ou indiretamente, seus clientes,

utilizando as facilidades de comunicação e de transferência de dados mediadas pela *internet*.

Diariamente são realizadas inúmeras transações comerciais na *internet*, e a tendência é que estas transações aumentem ainda mais, pois o aumento de usuários de computadores se conectando à *internet* é cada vez maior, sendo necessário proteger os dados transitados nas transações *on-line*. As assinaturas digitais podem fornecem esta segurança. O comércio eletrônico, ao aceitar a assinatura digital e se beneficiar com sua utilização, busca segurança em sua atividade. Por catalisar o crescimento da tecnologia em virtude da lógica inerente ao sistema capitalista, uma vez que as empresas “pontocom” se sintam mais seguras, a disponibilidade de serviços *on-line* deverá aumentar.

As transações comerciais eletrônicas são bastante semelhantes às transações comerciais tradicionais. Os produtos são apresentados no *site* da empresa, bem como os preços e as formas de pagamento e os prováveis compradores avaliam as opções, formas de pagamento e efetuam o pedido. Após confirmação da compra, a empresa envia o produto adquirido ao cliente.

Segundo Volpi (2001, p.33) para garantir a fidelidade, tanto da empresa quanto do comprador, o sistema da empresa é controlado na assinatura digital descrita em dois modelos:

- **Modelo Formalista**

No modelo formalista, não há preocupação quanto à segurança da assinatura e, tanto a tecnologia aplicada na assinatura como a legislação que a regulamenta seriam estáticas, é possível dizer que se faz necessário a confiança mútua entre a empresa e o comprador, não existindo um documento que ateste o contrato. Apesar do baixo custo, a aplicação deste modelo não se torna viável no contexto atual.

- **Modelo *risk-based***

Já no modelo *risk-based*, a preocupação com a segurança é o foco principal. A assinatura eletrônica requer autenticações específicas que evitem o repúdio da autoria, pois essas precisam ser dinâmicas, com nível de segurança

proporcional ao conteúdo da transação, garantindo a fidelidade e a autenticidade da ocorrência. O custo envolvido neste modelo aumenta consideravelmente em relação ao modelo formalista, e mesmo com o mesmo nível de segurança, dois documentos diferentes do mesmo remetente recebem assinaturas digitais distintas, uma vez que a assinatura é elaborada com base no conteúdo assinado.

2.3.2 Transações bancárias

Segundo relatado no *site* do Banco do Brasil S/A, a partir de novembro 2001, o governo brasileiro determinou que seja instituído o Sistema de Pagamentos Brasileiros (SPB), o qual exige que os bancos brasileiros utilizem a assinatura digital no processo de compensação bancária, com o intuito de garantir mais segurança e agilidade às transações e informações que circulam pela *Internet* entre o Banco Central e as outras instituições financeiras. Essa determinação se deu após o Governo ter definido as regras de atuação da Infra-estrutura de Chaves Públicas, ou seja, a tecnologia que vai gerar as Assinaturas Digitais. É com base nesse dispositivo que alguns bancos comerciais vêm implementando sistemas de assinatura eletrônica, utilizando senhas ou mesmo certificados digitais (os certificados digitais serão abordados no capitulo 2.4 desta dissertação).

Com o objetivo de proteger o cliente e estimular as transações *on-line*, os bancos estão lançando programas de certificação digital, buscando aumentar a segurança das transações bancárias feitas via *Internet*. A estratégia dessas instituições é de atingir contas corporativas e seus prepostos, funcionários dos próprios bancos e demais correntistas.

Uma das vantagens da certificação digital é que, após a sua emissão, o correntista passa a assinar digitalmente as suas transações, podendo efetuar qualquer tipo de transação enquanto estiver conectado à *Internet* (CERTISIGN, 2003).

Os clientes devem fazer sua adesão à Certificação Digital oferecida pelos bancos imprimindo o termo de adesão, obtido no *site* da instituição, ou dirigindo-se pessoalmente à agência bancária. Em ambos os casos, a liberação do Certificado

para uso na *Internet* só acontece após a conferência da assinatura manual dos clientes, nas agências detentoras das contas. Após a conferência da assinatura, é enviada ao cliente uma mensagem eletrônica, avisando-o da liberação para uso.

O Regulamento constitui-se na utilização do atributo de segurança, denominado Certificado Digital, que identifica o cliente e promove a autenticidade de suas transações via *Internet*, junto aos bancos. O certificado digital irá conter: a chave pública de criptografia, data de validade da chave pública de criptografia, nome e assinatura digital da entidade certificadora, o número de série do certificado digital, emissão do certificado digital e os dados cadastrais do cliente (CERTISIGN, 2003).

A validade do certificado é de doze meses, contados a partir da data de sua emissão, a qual estará registrada, juntamente com a data de seu vencimento, no próprio certificado digital do cliente; sua revogação poderá ser feita a pedido do cliente e a qualquer tempo.

O certificado digital será revogado quando: o sigilo da senha pessoal de acesso tenha sido violado, houver perda, destruição ou impossibilidade de recuperação do certificado digital e, em último caso, quando não houver mais interesse em utilizar o certificado.

Após a adesão à certificação digital, o cliente deverá, obrigatoriamente, utilizá-la para acesso aos produtos e serviços do banco disponibilizados via *Internet*. Em qualquer tempo, o cliente poderá copiar o seu certificado digital do computador onde estiver instalado, para outros computadores ou qualquer outra mídia, conferindo portabilidade ao certificado.

Por sua vez, os bancos devem garantir a impossibilidade de emissão de certificados digitais com número de série idêntico ao de qualquer outro certificado que tenha sido emitido sob as regras firmadas em contrato do banco com a entidade certificadora. Além disso, devem disponibilizar, via *Internet*, orientações e meios para que o cliente possa solicitar a emissão, revogação e renovação do seu certificado digital.

Os bancos poderão, a qualquer tempo, introduzir modificações no regulamento, mediante prévia e tempestiva divulgação aos clientes e registro em Cartório.

O regulamento dos bancos deverá estar registrado em Cartório de Registro de Documentos da cidade de Brasília DF e arquivado em cópia microfilmada.

No que diz respeito aos clientes, estes deverão requerer proteção especial do tipo *antivírus* (programas que detectam e eliminam o vírus de computador), *antihackers* (programas que dificultam ou impedem pessoas mal intencionadas invadir os arquivos armazenados no computador), *antitrojans* (programas que detectam e eliminam *trojans* – cavalo de tróia – do computador), sendo de responsabilidade do cliente manter tais programas atualizados no seu computador.

2.3.3 Atos jurídicos

Desde que a *Internet* se tornou um meio eficaz de comunicação entre pessoas físicas e jurídicas, a questão da segurança tornou-se presente como elemento garantidor do sucesso dessas atividades e, em função deste elemento, ressurgiram os modos de cifrar as mensagens, de forma que apenas o remetente e o receptor pudessem ter acesso ao teor dos documentos envolvidos na transação, garantindo o sucesso dessas relações. Desta forma, uma das mais importantes transformações, na área jurídica, se deu na conceituação de Documento.

A expressão "documento", sempre esteve atrelada à idéia de um escrito oficial que identifica uma pessoa. No meio jurídico, representa um escrito que faz fé daquilo que atesta, de forma que, se apresentado em juízo, prova o que o litigante alega (LUCCA E SIMÃO, 2000).

Com o advento da *Internet* e a sua rápida expansão pelo mundo, o conceito de documento teve que passar por uma adequação, de forma a se tornar viável a sua aplicação ao meio virtual, tendendo a alcançar os mesmos objetivos já consolidados no meio tradicional.

Dentro dessa nova conjuntura, surgiu, então, a conceituação do Documento Eletrônico, que guarda as principais características do Documento Tradicional, excetuando-se o meio no qual é celebrada uma transação e a questão atinente à identificação da pessoa. É possível conceituar o Documento Eletrônico como sendo a representação não material de um fato, tendente a alcançar segurança jurídica.

Criando um paralelo entre as duas conceituações, é possível dizer que a principal diferença dos documentos tradicionais com os documentos eletrônicos é que os primeiros são representados por escritos em papéis e os segundos por bits.

É unanimemente reconhecida a necessidade de se dar eficácia e validade jurídica aos contatos virtuais, de modo que possam ser equiparados aos documentos conhecidos atualmente, que estão ligados a um meio material tangível.

Entre os diversos novos recursos que foram colocados à disposição dos usuários de computadores, a comunicação eletrônica tem recebido a adesão da quase totalidade dos advogados, substituindo em grande parte a comunicação que antes era feita pelo correio através de carta ou pelo *fax*.

A comunicação eletrônica decorre das inúmeras vantagens em relação à comunicação tradicional. Burnett e Paine (2002, p.251), descrevem as principais vantagens, nos seguintes tópicos:

- **Integridade de mensagem:** uma assinatura digital é superior a uma assinatura escrita à mão, pois ela atesta o conteúdo de uma mensagem e a entidade do assinante.
- **Economia:** o uso de sistemas abertos (como a *Internet*) como mídia de transporte pode ajudar consideravelmente a economizar tempo e dinheiro. Além de adicionar automação significativa, possibilita que os dados sejam assinados digitalmente e enviados de uma maneira oportuna.
- **Armazenamento:** os dados de negócios (contratos e documentos semelhantes) podem ser armazenados muito mais facilmente em uma forma eletrônica do que na forma de papel. Um documento que foi assinado digitalmente pode ter validade indefinida, devido ao registro de data/hora que permite provar a validade de um contrato mesmo se, em algum momento, a chave do assinante for comprometida depois que o contrato já estiver assinado.
- **Diminuição de erros:** Se adequadamente implementadas, as assinaturas digitais reduzem o risco de fraude e tentativa, por uma parte, de repudiar (negar) o contrato.

Vários estudos estão sendo feitos por diversos organismos ligados à administração da Justiça, visando dar os primeiros passos para a criação do “processo virtual”, cujo objetivo consiste basicamente em possibilitar que o processo

seja desenvolvido totalmente a partir de documentos eletrônicos, muitos deles enviados por meio de comunicação eletrônica (KAMINSKI, 2002). Desta forma, a comunicação eletrônica constitui um instrumento cada vez mais importante para o exercício da advocacia, tanto na comunicação entre o advogado e o cliente, quanto na evolução da comunicação entre o advogado e os órgãos do poder público.

As assinaturas digitais permitem a realização de inúmeros atos jurídicos à longa distância, possibilitando uma enorme flexibilização das relações que necessitam serem concebidas de maneira expressa. Esta capacidade precisa estar devidamente regulamentada para que possa gozar de total confiança por parte daqueles que a utilizarão (DINIZ,1999).

Um exemplo da aplicabilidade deste sistema pode ser verificado, no Brasil, analisando o funcionamento do Governo Federal, que vem utilizando o sistema de transmissão eletrônica de atos normativos entre os ministérios e a Presidência da República. A partir da implantação desse sistema automatizado e em tempo real, com toda tramitação acontecendo economicamente via *Internet*, a agilidade da administração federal só vem aumentando. Os documentos recebem a assinatura digital dos ministros e são enviados para conhecimento do Presidente.

Em outros países, isto também já vem acontecendo. Na França, por exemplo, o Governo utiliza sistema semelhante a esse, mas a tramitação dos documentos se dá na rede interna deste Intranet, que só é acessada em computadores de órgãos públicos. No Brasil, a *Intranet* Governamental ainda está em processo de implantação.

Em fevereiro de 2002, a Ordem dos Advogados do Brasil (OAB)-SP começou a operar sua certificadora em fase de testes públicos e os advogados de São Paulo puderam testar a assinatura digital. Em Abril de 2002, o Conselho Federal desse órgão aprovou o Provimento que institui o ICP-OAB, o que tornou oficial a certificação digital dos advogados.

O projeto ICP-OAB vem sendo desenvolvido desde o ano 2000, porque a própria OAB preferiu ser a entidade certificadora dos advogados e, para tanto, precisou desenvolver seu próprio sistema para garantir a integridade das informações e restringir o acesso aos dados unicamente à Ordem. Inicialmente a implantação do sistema se dará no Conselho Federal, e, posteriormente, deverá ser expandida para todas as seccionais (KAMINSKI, 2002).

A principal vantagem da assinatura digital é a integridade da informação e a identificação virtual confiável, além de facilitar o trabalho dos profissionais, que poderão fazer tudo eletronicamente. Um advogado que precisar peticionar junto a um órgão de Brasília poderá fazê-lo à distância. Dessa forma, os advogados avançam de forma consistente na informatização do Judiciário.

2.4 Certificados Digitais e o padrão X.509

Como visto anteriormente, para realizar uma assinatura digital, é necessário um par de chaves, sendo uma chave privada e a outra pública interligadas através de cálculos matemáticos. O autor do documento utiliza a chave privada para realizar a assinatura, e o receptor utiliza a chave pública para verificá-la, tornando-se inevitável que o receptor do documento possua a chave pública, sendo esta gerada a partir da chave privada do emissor.

Este procedimento pode ocasionar um grande problema. Assim como qualquer outro dado trafegado em uma rede de computador, a chave pública também pode ser susceptível à manipulação durante o trânsito. Desta forma, um possível interceptador poderia alterar o documento original e até mesmo falsificar a assinatura digital.

Em um pequeno grupo de usuários confiáveis, a distribuição da chave pública poderia ser feita pelo método manual. Entretanto, em uma população geograficamente dispersa, este método torna-se inviável. Por esta razão, a solução foi a criação dos certificados digitais, que podem ser definidos como documentos eletrônicos assinados digitalmente por um terceira parte confiável, que associam o nome de uma pessoa ou instituição a uma chave pública criptográfica. (GARFINKEL E SPAFFORD, 1999).

Utilizando novamente o mundo real do papel e caneta, por medida de comparação, as entidades confiáveis conhecidas como autoridades certificadoras (*Certification Authorities* - CAs), seriam os cartórios, sendo que a certificação digital seria um serviço notarial efetuado por um tabelião. As CAs podem ser um

departamento de segurança da empresa, um governo ou uma companhia privada que trabalhe com a emissão de certificados para usuários da *Internet*.

Os certificados de chave pública (*public-key certification* – PKC), como são conhecidos, por certificarem a assinatura digital, formam um conjunto de dados à prova de falsificação, que atesta a associação de uma chave pública a um usuário final através das CAs, sendo um meio seguro de distribuir para as partes verificadoras as chaves públicas dentro de uma rede. (BURNETT E PAINE, 2002).

Vários certificados estão em utilização atualmente. Dentre eles, o *Pretty Good Privacy (PGP*) que é patenteado, e os certificados populares específicos de um aplicativo, como o de Transação Eletrônica Segura (*Secure Eletronic Transactions* – SET) e o Protocolo de segurança de *Internet* (*Internet Protocol Security* – IPSec). No entanto, o mais amplamente aceito é o X.509 – Versão 3, da *INTERNATIONAL TELECOMMUNICATIONS UNION* (KUROSE E ROSS, 2003).

O padrão X.509 faz parte de um conjunto de recomendações do X.500, que define um serviço de diretório. Publicado em 1988, o X.509 está atualmente na sua terceira versão.

As alterações de versão no padrão X.509 foram realizadas pelos seguintes fatos:

- **X.509v1:** Devido ao pequeno número de campo que limitava a sua utilização. Além disso, alguns problemas de segurança também foram identificados.
- **X.509v2:** Nesta versão foram adicionados novos campos, com o objetivo de possibilitar a reutilização de nomes iguais em diferentes certificados digitais, conforme ilustrado na figura 13 abaixo. Porém, ainda era deficiente em vários aspectos.
- **X.509v3:** Em resposta às deficiências encontradas nas duas versões anteriores, nesta versão foi adicionado o campo de extensão, o que torna um certificado mais flexível e com um leque de utilização muito maior.

Um certificado X.509, em sua terceira versão, certifica se uma chave pública foi assinada por uma instituição em particular. Essa certificação é selada por meio do uso de uma assinatura digital. (GARFINKEL E SPAFFORD,1999)

Burnett e Paine (2002, p.146) afirma que todas as versões dos certificados X.509 contêm os seguintes campos:

- **Versão (*Version*):** Diferencia as sucessivas versões do certificado, permitindo possíveis versões futuras;
- **Número serial do certificado (*Certificate Serial Number*):** Contém um valor inteiro único para cada certificado e é gerado pela CA;
- **Identificador do algoritmo de assinatura (*Signature Algorithm Identifier*):** Identifica o indicador do algoritmo utilizado para assinar o certificado, juntamente com quaisquer parâmetros associados;
- **Nome do emissor (*Issuer N*ame):** Identifica o nome distinto pelo qual a CA cria e assina aquele certificado;
- **Validade – Não antes / Não depois (*Validity – Not before/After*):** Contêm dois valores de data/hora – que definem o intervalo temporal no qual um certificado pode ser considerado válido a menos que, caso contrário, seja revogado;
- **Nome do sujeito (*Subject Name*):** Identifica o nome distinto da entidade final a que o certificado se refere, isto é, o sujeito que mantêm a chave privada correspondente. Este campo deve ter uma entrada, a menos que um nome alternativo seja utilizado nas extensões da versão três;
- **Informações sobre a chave pública do sujeito (*Subject Public Key Information*):** Contêm o valor da chave pública do sujeito, bem como o identificador de algoritmo de quaisquer parâmetros associados ao algoritmo pelos quais a chave deve ser utilizada. Este campo sempre deve ter uma entrada.

A figura 14, ilustra a estrutura dos certificados X.509 e seus respectivos campos, de acordo com cada versão.

A sintaxe dos certificados que seguem o padrão X.509 é expressa utilizando uma notação especial conhecida como Sintaxe Abstrata de Anotação Um(*Abstract Syntax Notation One* – ANS.1), que originalmente foi criada pela *Open System Interconnecton* – OSI, para ser utilizada em vários protocolos X.500. A ASN.1 introduz a definição de Identificador de Objetos (*OBJECT IDENTIFIERES* – OIDs) e torna possível a comunicação da semântica da informação entre sistemas distintos.

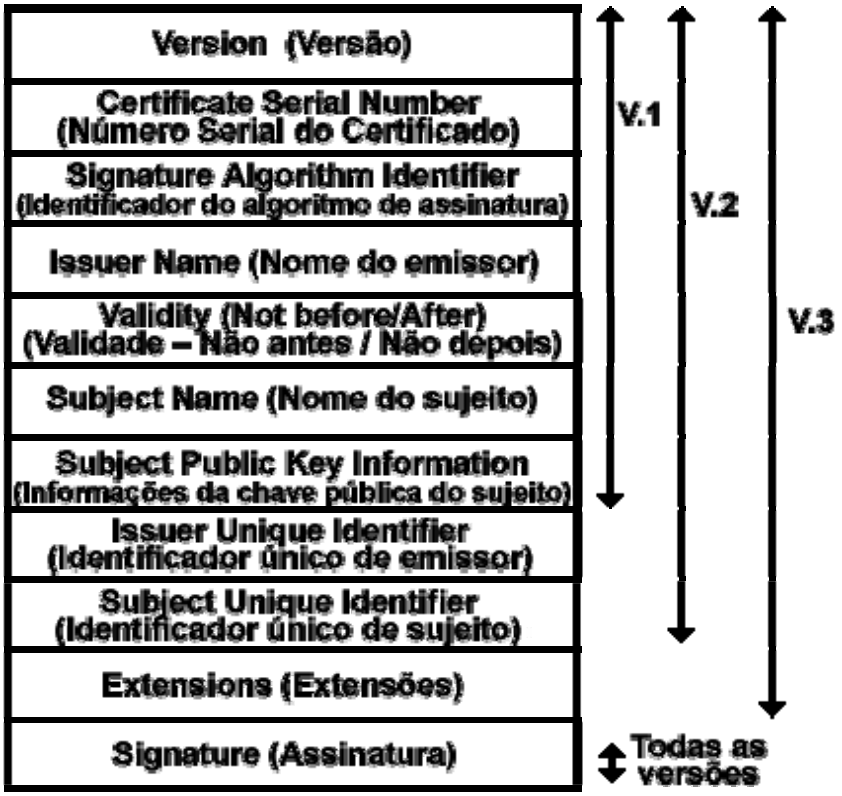

FIGURA 14 – Estrutura do certificado X.509.
FONTE: BURNETT, Steve; PAINE, Stephe. (2002, p.147).

2.4.1 Infra-estrutura de chave pública

Conforme já mencionado, as CAs servem como terceiros confiáveis, emitindo certificados que contêm uma chave pública, o nome do usuário e outros informações de identificação. Estes certificados, quando assinados por uma CA, são armazenados em diretórios públicos, podendo ser recuperados para verificar uma assinatura, ou encriptar documentos.

Uma infra-estrutura de chave pública (*PUBLIC KEY INFRASTRUCTURE* – PKI) tem como fundamento definir uma arquitetura que permite reunir um conjunto de elementos, orientá-los pelas mesmas regras e colocá-los em interação para um objetivo comum, desenvolvendo um ambiente seguro, cujos serviços apresentados sejam baseados em técnicas e conceitos relativos ao uso de criptografia assimétrica. (ADAMS E LLOYD, 1999).

2.4.1.1 Componentes de uma PKI

A infra-estrutura de uma PKI é composta por um conjunto de elementos, cada um com funções especificas que, interligados, permitem realizar o objetivo da PKI. Cada componente será descrito a seguir:

- **Usuário final**

Este usuário tem o intuito de realizar uma certificação digital. Podendo ser pessoa física ou jurídica.

- **Autoridade de Certificação**

Uma Autoridade de Certificação (*CERTIFICATION AUTHORITY* – CA) é a base de confiança de toda a PKI. Toda a confiança na infra-estrutura depende da sua assinatura.

A função de uma CA é gerar e fornecer os meios técnicos para a geração dos pares de chaves e emitir certificados digitais, com o intuito de realizar a recepção de pedidos de certificação e de revogação feitos pela Autoridade de Registro, e o retorno de certificados e de listas de certificados revogados, respectivamente.

Uma CA deve fornecer sua própria chave pública para todos os usuários finais certificados e para todas as partes verificadoras que possam utilizar as informações certificadas.

As CAs podem pertencer a duas categorias distintas:

- As CAs públicas, que operam via *Internet*, fornecendo serviços de certificação para o público em geral, certificando não apenas os usuários, mas também as empresas;
- As CAs privadas, que normalmente são encontradas dentro de uma corporação ou rede similar, e tendem a licenciar apenas usuários finais dentro da própria população, fornecendo assim, controles de acesso e autenticação mais rigorosos. (Northcutt, et al, 2002).

Segundo Volpi (2001, p.36), as principais informações na emissão de um certificado são:

- Chave pública do autor.
- Nome e endereço de e-mail do autor.
- Data de validade da chave pública.
- Nome da autoridade certificadora que emitiu seu Certificado Digital.
- Número de série do certificado digital.
- Assinatura digital da autoridade certificadora.

No entanto, é importante observar que somente algumas informações básicas estão contidas em um certificado digital e que existem vários formatos utilizados por diferentes autoridades certificadoras.

- **Autoridade de Registro**

A Autoridade de Registro (*Registration Authority* - RA) é de uso facultativo, podendo servir como uma entidade intermediária entre uma CA e seus usuários finais.

Sua utilização é facultativa, pois seus serviços podem ser realizados pelas Cas. Porém, com o crescimento de usuários finais dentro de uma dada comunidade de PKI, também aumenta a carga de trabalho de uma CA, sendo aconselhável a divisão de tarefas. Esta divisão de tarefas tem como finalidade aliviar a carga funcional das CAs e repartir responsabilidades.

Segundo Burnett e Paine (2002, p.153), uma RA comumente fornece as seguintes funções:

- Aceita e verifica as informações de registro sobre novos registradores;
- Gera chaves em favor de usuários finais;
- Aceita e autoriza solicitações para uma cópia de segurança (*backup*) e recuperação de chaves;
- Aceita e autoriza solicitações para revogação de certificados;
- Distribui ou recupera dispositivos de *hardware* como *tokens*, quando necessário.

- **Repositório de certificado**

O Repositório de certificados (*Certificate Repository*) é o componente responsável por armazenar os certificados. Após a geração de um certificado, ele deve ser armazenado para ser utilizado posteriormente.

Não há padrão requerido de diretório. Alguns aplicativos, como *Lotus Notes* e *Microsoft Exchange*, utilizam diretórios patenteados. Atualmente, os diretórios baseados no padrão X.500 também estão ganhando popularidade. (GARFINKEL E SPAFFORD, 2000).

Os diretórios X.500 estão se tornando mais amplamente aceitos porque, além de atuarem como repositório de certificados, fornecem aos administradores uma localização central para entrada das informações de atributos pessoais. A acessibilidade às informações sobre o usuário é controlada a partir de diferentes clientes, utilizando, para tanto, um protocolo de acesso ao diretório como o *LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL* (LDAP), que foi projetado para fornecer aos aplicativos um meio para acessar os diretórios X.500. (BURNETT E PAINE, 2002).

- **Servidor de recuperação de chaves**

Este componente assegura a recuperação de chaves privadas perdidas devido a vários fatores, inclusive à perda de dados causada pela destruição da mídia na qual se encontrava guardada.

As chaves privadas armazenadas deverão ser as de cifra, e não as de assinatura, garantindo assim a propriedade de não repúdio. Desta forma, é de mera importância que nem a própria CA tenha acesso a chave privada de assinatura dos titulares, armazenando no Servidor de recuperação de chaves apenas a chave privada da cifra.

2.4.2 Revogação de certificado

Os certificados digitais são criados com a intenção de serem válidos por todo o tempo de vida estipulado no campo Validade (*Validity Not Before/ Not After*). Porém, em alguns casos, a chave privada poderá ser comprometida. Sendo assim, um certificado ainda em vigor não mais deverá ser utilizado. Desta forma, as CAs, responsáveis por emitirem os certificados, necessitam de uma maneira para revogar um certificado em vigor e notificar as partes verificadoras sobre a revogação. O método encontrado foi o da criação de uma lista de revogação de certificados (*Certificate Revocation List* – CRL) (SCHNEIER, 2001).

Em sua forma básica, uma CRL é uma estrutura de dados assinada, contendo uma lista de data/hora dos certificados revogados. Após criar uma CRL e assiná-la digitalmente, ela pode ser distribuída livremente através de uma rede, ou armazenada em um diretório, da mesma forma que os certificados são tratados (BURNETT E PAINE, 2002).

Uma das formas de não revogar muitos certificados, é limitar um tempo bastante curto no qual eles possam ser utilizados. Quanto menor o tempo, maior a possibilidade do certificado não ser revogado (SCHNEIER,2001).

2.5 Processos legislativos

De acordo com os estudos realizados, são notáveis os benefícios que a assinatura digital pode proporcionar para uma comunidade que utiliza cada vez mais os meios eletrônicos, no dia-a-dia, para se comunicar, trocar documentos digitais ou até mesmo realizar algum tipo de negócio, utilizando as redes de computadores. Portanto, para que a assinatura digital se torne mais prevalecente, é de fundamental importância que se ofereça a um documento assinado digitalmente o mesmo status jurídico dos documentos em papel, fornecendo um método confiável, seguro e sancionado juridicamente, para que a assinatura digital elimine a necessidade de assinar documentos em papel. Desta forma, encorajando e facilitando o comércio

eletrônico, repartições públicas e qualquer entidade que necessite de certificar a autoria e a validade de um documento digitalizado. Para que isso ocorra é necessário ter leis a favor da assinatura digital, garantindo os mesmos direitos de uma assinatura em papel. (BURNETT E PAINE 2002).

A organização que representa a profissão jurídica nos Estados Unidos (*American Bar Association* – ABA) tem feito um trabalho considerável relacionado aos aspectos jurídicos das assinaturas digitais. Em 1996, o *INFORMATION SECURITY COMMITTEE*, *SECTION OF SCIENCE AND TECHNOLOGY* publicou um documento intitulado "*DIGITAL SIGNATURE GUIDELINES*" – Diretrizes da Assinatura Digital.

Estas diretrizes foram projetadas para fornecer declarações abstratas e gerais de princípios destinadas a servir como uma base unificadora em longo prazo de uma lei para a assinatura digital em diversos documentos legais. Após a divulgação deste documento, vários estados escolheram modelar sua própria legislação sobre assinatura digital.

2.5.1 Conceitos jurídicos das assinaturas digitais

A facilidade para se copiar e alterar um documento eletrônico sem que alguém detecte é muito grande, além disso, diferentemente de textos escritos a mão, os textos codificados digitalmente não são únicos. A assinatura de um documento eletrônico não está fisicamente relacionada ao conteúdo do documento. Para que os documentos eletrônicos, contendo uma assinatura digital, tenham a mesma validade jurídica dos documentos que possuem uma assinatura comum, torna-se necessário provar para um terceiro imparcial (um foro, um juiz ou um mediador), que o conteúdo do documento eletrônico é genuíno e que foi originado pelo remetente. É necessário ainda que o remetente não possa negar futuramente o conteúdo do documento no qual consta a sua assinatura digital, sendo de fundamental importância aplicar os conceitos de não-repúdio e autenticação. Segundo Burnett e Paine (2002), O conceito de não-repúdio pode ser dividido em três tipos (não repúdio de origem, não repúdio de entrada e não repúdio de submissão), e o serviço de autenticação com

relação a assinaturas digitais devem ser entendidos em dois tipos (autenticação do assinante e autenticação de dados).

- **Não-repúdio de origem:** protege o destinatário de uma comunicação, garantindo a identidade de quem a originou. Confirma ainda a data/hora em que a mensagem foi enviada e a sua integridade (garante que o conteúdo da mensagem não foi modificado a transmissão).
- **Não-repúdio de entrega:** protege o remetente de uma comunicação, garantindo a identidade de quem a originou e também confirma a data/hora em que a mensagem foi enviada e a sua integridade.
- **Não-repúdio de submissão:** é semelhante ao não-repúdio de origem e de entrega, exceto por ele ser utilizado para proteger o remetente contra qualquer reivindicação feita pelo destinatário de que os dados não foram enviados ou não foram enviados em uma data/hora específica.
- **A autenticação de assinante:** garante que o assinante tenha ciência do conteúdo do documento que está assinando, para que o mesmo possa ter poder legal. Essa assinatura, além de indicar quem assinou, impossibilita uma outra pessoa de reproduzir a assinatura sem uma autorização.
- **A autenticação de dados:** garante que nenhuma modificação futura possa ser feita em um documento após o mesmo já ter sido assinado. Normalmente, a autenticação de dados é acompanhada pela "autenticação de origem de dados", a qual associa uma pessoa real a um documento específico.

Outro conceito relevante é o denominador "data/hora" que, segundo Schneier (2001, p.232), "é um conjunto de técnicas que permite determinar se um documento foi criado ou assinado em uma determinada (ou antes) data/hora".

Na maioria das vezes, os sistemas de registro de data/hora utilizam um terceiro confiável, chamado de autoridade registradora de data/hora (*TIME-STAMPING AUTHORITY* – TSA). Esse serviço é de grande importância no que diz respeito aos conceitos jurídicos relacionados às assinaturas digitais, possibilitando uma autenticação a longo prazo de documentos digitais. Por exemplo, se o assinante de um documento vier a revogar seu certificado após tê-lo assinado, é

possível provar que o mesmo foi assinado antes que a chave correspondente da assinatura fosse revogada.

2.5.2 Patentes dos Algoritmos

As patentes aplicadas aos programas de computador são conhecidas como “patentes de *software*”. Algumas das mais antigas patentes de *software* concedidas pelo Departamento de Marcas e Patentes dos Estados Unidos são relacionadas à criptografia.

Garfinkel e Spafford (1999) relatam que: “embora se acreditasse não ser possível patentear algoritmos computacionais, as patentes de criptografia foram aceitas, pois eram parentes de dispositivos criptográficos internos de *hardware*”.

O sistema RSA de criptografia de chave pública é utilizado em diversos sistemas criptográficos como PGP, S/MIME, SSL, entre outros.(ABRANTES, 2002). Em setembro de 2000, a patente do algoritmo RSA expirou e, atualmente, qualquer firma ou indivíduo pode criar implementações desse algoritmo (BURNETT E PAINE, 2002). Apesar do algoritmo RSA ser amplamente recomendado e aceito pela comunidade criptográfica, ainda existem restrições quanto à sua utilização em determinados países.

A tecnologia de chave pública é largamente adotada na Europa, onde é possível encontrar cartões telefônicos inteligentes, com dispositivos criptográficos e ampla disseminação dos algoritmos de chave pública. Isto se deve ao fato de que, no Japão e na Europa, o inventor perde o direito à patente após a primeira revelação. Já nos Estados Unidos, o inventor tem um período de carência de um ano entre a primeira revelação pública da descoberta e o pedido de patente.

Os algoritmos de criptografia simétrica RC2 e RC4, também desenvolvidos pela RSA Data Security, não foram objeto de patente, mas sim mantidos como segredo comercial. No DES e Triple DES não se aplica patente por serem padrão da NITS - (*National Institute of Standards and Technology*) - (OAKS, 1999).

O algoritmo DSA, por ser um algoritmo padrão de assinatura digital e ter licença livre, pode ser utilizado por qualquer pessoa, em qualquer país, pois,

conforme relatado anteriormente, este algoritmo não criptografa os dados e sua finalidade se restringe às assinaturas digitais.

2.5.3 O controle sobre a criptografia

Devido à sua origem militar, e por sua função estratégica, a criptografia é altamente regulamentada em diversos países. As restrições quanto à importação e exportação da criptografia estão diretamente ligadas ao tamanho das chaves criptográficas (OAKS, 1999).

França e Israel, por exemplo, exercem forte controle, não somente sobre a importação e exportação dessa tecnologia, mas também sobre sua utilização doméstica. O controle francês vê a criptografia acima de 40 bits como elemento crítico para a defesa nacional; assim, o tamanho máximo liberado para uso público não passaria de 56 bits, incapaz de garantir plena segurança aos dados que trafegam em redes abertas, como a *Internet*.

Em Israel, a importação, exportação, produção ou uso de qualquer produto de criptografia exige licença do Ministério da Defesa.

A China, a exemplo dos dois países já citados, também exerce forte controle sobre os procedimentos criptográficos. A importação ou exportação desse tipo de produto exige licença governamental, tanto para uso doméstico como empresarial.

Na outra ponta dessa tendência de restrições à produção e uso dos sistemas criptográficos, estão países como Canadá, Finlândia, Alemanha e Japão, que consideram os sistemas criptográficos necessários para a proteção de dados pessoais, o desenvolvimento do comércio eletrônico e de todos os negócios confidenciais. Por isso, existe, nesses países, uma grande preocupação com os aspectos criminais que, eventualmente, poderão advir da disseminação da criptografia, ficando as autoridades responsáveis por monitorar o desenvolvimento da tecnologia e pela aplicação das leis que regulamentam essas atividades (LUCCA E SIMÃO, 2000).

Na América Latina, inclusive no Brasil, essa questão não tem merecido maiores preocupações dos setores oficiais, apesar de haver um consenso sobre o

fato de a criptografia ser uma importante ferramenta para a privacidade do cidadão em um mundo cada vez mais informatizado.

Nos Estados Unidos, a criptografia é considerada artigo de defesa e faz parte da Lista de Munições dos Estados Unidos, estando relacionada na categoria Equipamento Militar Auxiliar. Mesmo assim, não há restrições para seu uso doméstico.

Segundo Burnett e Paine (2002), em janeiro de 2000, o Governo dos Estados Unidos anunciou um relaxamento significativo quanto às restrições para a exportação de criptografia forte, permitindo assim que empresas norte-americanas pudessem competir mundialmente no setor de criptografia.

Para uma criptografia ser considerada forte, ela tem que mudar pelo menos 50% do resultado final, ao ser alterado um caractere no texto original ou o mínimo possível na chave de criptografia. Segundo Schneier (2001), para a criptografia simétrica um tamanho mínimo recomendado é de 90 bits e para a assimétrica o tamanho é de 1.024 bits.

2.5.4 Regulamentação da assinatura digital

Há algum tempo a legislação da assinatura digital tem sido uma questão constante no mundo. Isso decorre do número crescente de transações comerciais internacionais, habitualmente conhecido como comércio eletrônico.

A primeira iniciativa, em legislação, sobre a assinatura eletrônica ocorreu no ano de 1995, no estado de Utah (Estados Unidos), onde foi sancionada a primeira lei estadual referente à assinatura digital, que ganhou notoriedade na arena jurídica.

Em março de 1996 ela foi retificada e é amplamente reconhecida como o primeiro passo no reconhecimento jurídico da tecnologia de assinatura digital. O objetivo principal da legislação da assinatura digital do estado de Utah é de promover o desenvolvimento de uma infra-estrutura de chave-pública.

Na legislação vigente no estado de Utah são detalhados os direitos e as responsabilidades das partes em uma transação que utilize a criptografia de chave pública. O Estado oferece a licença de Autoridades certificadoras pelo Utah

Department of Commerce, e as CAs que obtêm a licenças são tratadas com regras de responsabilidade favoráveis (BERNSTEIN et al, 1997).

Outros estados norte-americanos começaram a levar em consideração leis modeladas de acordo com a lei de Utah, como, por exemplo, os estados de Washington e Geórgia. No entanto, mais tarde, esses e alguns outros estados permitiram que os projetos de lei morressem, optando por um estudo mais aprofundado. Outros estados adotaram métodos menos reguladores e específicos de tecnologia. Burnett e Paine (2002, p.259) exemplificam a aplicação das assinaturas digitais nesses estados da seguinte forma:

> A Califórnia e o Arizona sancionaram uma legislação permitindo o uso de assinaturas digitais em transações que envolvem entidades estaduais. Essa legislação autorizou os dois secretários desses estados a promulgar regulamentos para alcançar o propósito de tal decreto. Outros ainda aprovaram leis permitindo o uso de assinaturas digitais para propósitos específicos como registros médicos ou para propósitos de orçamento e contabilidade como a verificação de assinatura digital pela secretaria da fazenda.

A importância de se regularizar a assinatura digital tornou-se ainda mais expressiva quando, em 1996, a União Européia adotou a lei da Comissão das Nações Unidas sobre o Direito do Comércio Internacional (*United Nations Commission on International Trade Laic* - UNCITRAL), que se tornou uma lei modelo sobre o comércio eletrônico, que serviria de referencial aos países-membro (Volpi, 2001).

Segundo Burnett e Paine (2002, p.259), “a lei modelo UNCITRAL é de alto nível, permitindo um método para as assinaturas e registros eletrônicos sem nenhuma menção quanto às assinaturas digitais ou à criptografia”.

No que se refere especificamente à atuação da assinatura digital, a lei modelo descreve que, quando uma lei requisitar a assinatura de uma pessoa, este requisito será considerado preenchido por uma mensagem eletrônica quando:

- for utilizado algum método para identificar a pessoa e indicar sua aprovação para a informação contida na mensagem eletrônica;

- e tal método seja tão confiável quanto seja apropriado para os propósitos para os quais a mensagem foi gerada ou comunicada, levando-se em consideração todas as circunstâncias do caso. Incluindo qualquer acordo das partes a respeito.

Volpi (2001, p. 46) declara que:

> Quanto às leis destinadas a regulamentar situações que envolvam aspectos tecnológicos, devem restringir-se somente aos princípios que norteiam esta tecnologia, pois, de outra forma, corre-se o risco de criar verdadeiros entraves legais para a evolução da matéria.

Na Europa, as leis para assinaturas digitais estão em vigor há mais de uma década. A Alemanha já tem a sua *Informations Und Kommunikationsdienste Gesetz Lukdg*, Lei Federal que estabelece condições gerais para o uso das assinaturas digitais, quanto a seu aspecto de segurança que se baseia no sistema de Criptografia (LUCCA E SIMÃO, 2000).

E assim, outros países, como a Itália e a Bélgica adotaram procedimentos semelhantes. A ONU, por meio da comissão chamada UNCITRAL (Comissão das Nações Unidas sobre o Direito do Comércio Internacional) já volta os seus olhos para essa questão da segurança nas relações cibernéticas e reconhece os certificados emitidos por uma entidade certificadora de outro Estado membro da União Européia, se este possuir um grau de segurança equivalente ao dos países membros da ONU (ALBERTIN, 1999).

A Lei de Assinaturas Eletrônicas no Comércio Nacional e Global (*Eletronic Signatures in Global And National Commerce* - E-SIGN) é um projeto de lei para assinaturas eletrônicas, criado após longas negociações entre o Senado e o Congresso dos Estados Unidos.

O *E-SIGN Act* foi assinado pelo ex-presidente Bill Clinton, em 30 de junho de 2000, tornando-se lei efetiva em 1° de outubro de 2000 (NORTHCUTT, 2002). Foi originalmente projetada para impulsionar o comércio eletrônico da *Internet*, eliminando trabalho em papel que advém de contratos. Essa legislação dá reconhecimento e efetiva juridicamente as assinaturas.

A legislação denominada E-SIGN fornece o importante fundamento para a viabilidade e aceitação legal de assinaturas digitais e tecnologias de comércio

eletrônico associadas, em todas as regiões dos Estados Unidos. Os princípios estipulados na E-SIGN, oferecem a arquitetura necessária para expandir a adoção dessas tecnologias, assim como de diretrizes legais para sua utilização.

Baseados na lei E-SIGN, alguns serviços foram projetados para o suporte das assinaturas digitais, serviços de autoridades certificadoras, registro de data/hora e de tabelião digital, tais como:

- **VeriSign –** conforme já relatado, é uma CA que emite certificados de chave pública aos usuários finais em qualquer lugar do mundo (KUROSE E ROSS, 2003).
- **CertSign** – oferece os mesmos serviços que a VeriSign, por ser a sua única afiliada brasileira.
- **DigSign –** é uma empresa que já começou a vender os seus serviços digitais de registro de data/hora e de tabelião (BURNETT E PAINE, 2002, p.265).

Os profissionais da área jurídica devem se preparar para se tornarem tecnicamente bem informados. E espera-se que esse número aumente à medida que outros casos jurídicos relacionados começarem a emergir no futuro (GARFINKEL E SPAFFORD,1999).

2.5.5 Regulamentação da assinatura digital no Brasil

A ciência do Direito é, por natureza, uma ciência social, porém, paradoxalmente, conservadora. Isso implica grande dificuldade de se assimilar prontamente as inovações sociais, principalmente no que diz respeito à autenticação eletrônica e à aplicação da assinatura digital em documentos (contratos, certidões, entre outros), que possam ter valor jurídico mediante a utilização de tal tecnologia.

Se, no passado, havia tempo adequado para que o Direito identificasse as modificações nos hábitos e costumes da sociedade, e, ainda assim, havia certa dificuldade em acompanhá-las, atualmente, em plena era da informação, a dificuldade é incontestavelmente maior.(BITTENCOURT,2002).

No entanto, devido à rapidez dos avanços tecnológicos atuais e das inúmeras e diversas transações efetuadas utilizando-se a *Internet* como meio, a integração dos profissionais ligados à tecnologia de informação, à matemática e à área jurídica torna-se de suma importância para que as facilidades oferecidas possam ser aproveitadas da melhor forma pela sociedade em geral, garantindo segurança e respaldo jurídico à ela.

Atualmente, no Brasil, existem vários estudos que visam a regulamentação da assinatura digital e alguns projetos que buscam abordar a matéria de forma mais precisa já estão tramitando no Congresso Nacional. (LUCCA E SIMÃO, 2000).

A Normalização da assinatura digital se efetivou legalmente no Brasil, com a criação de um regulamento para a sua utilização pelo Poder Executivo Federal.

Com o intuito de viabilizar a circulação de documentos e informações entre os órgãos do governo, com maior garantia de segurança e confiabilidade, o Poder Executivo Brasileiro criou, em 5 de setembro de 2000, o Decreto N.º 3.587, regulando o uso de sistemas de assinaturas digitais pelas autoridades governamentais, do primeiro ao terceiro escalão do governo.

Apesar de ter sido um primeiro passo para a aprovação dos demais projetos que se encontram em tramitação, o decreto N° 3.587 está direcionado para uma tecnologia específica, o que não possibilita uma eventual flexibilização legal quando da ocorrência da evolução no meio tecnológico.

Alguns pontos relevantes do decreto em questão podem ser destacados, tais como, o ICP-Gov (Infra-estrutura de Chaves Públicas do Poder Executivo Federal). Trata-se de um conjunto de especificações (arquitetura, organização, técnicas, práticas e procedimentos) que visa à operacionalização de um sistema de certificação. Essa operacionalização é restrita ao método de cifragem assimétrica.

Dentro do ICP-Gov, criou-se a figura de uma AGP (Autoridade de Gerência de Políticas), organismo destinado a estabelecer os padrões (técnicos, operacionais e de segurança) a serem seguidos pelas CAs que seriam ligadas ao próprio ICP-Gov.

Cabe à AGP a criação de uma Autoridade Certificadora Raiz (AC Raiz), responsável pela emissão e manutenção dos certificados das CAs de órgãos da Administração Pública Federal e das CAs privadas credenciadas, bem como o gerenciamento da Lista de Certificados Revogados (CRL) (Certisign, 2003).

As RAs (Autoridades de Registro) também devem cumprir com suas responsabilidades em relação às CAs, responsabilidades estas já descritas anteriormente.

Um fator a ser considerado é que as CAs podem receber seu credenciamento em níveis diferenciados, de acordo com a finalidade na qual se proponham a atuar. No entanto as mesmas devem seguir critérios estabelecidos pela AGP. Esses padrões incluem aspectos técnicos internacionalmente reconhecidos e aspectos adicionais, tais como: plano de contingência, política e plano de segurança, análise de riscos, capacidade financeira da proponente e histórico no mercado. Em se tratando de uma CA privada, devem ser avaliados, ainda, antecedentes e histórico no mercado, além da cobertura jurídica e do seguro contra danos que a mesma fornece aos seus usuários.

Em relação ao decreto N° 3.587 Volpi (2001, p. 48) declarando que:

> Mesmo sendo uma regulamentação direcionada ao ambiente interno do governo, esse Decreto tem uma grande virtude no que tange à abertura de uma nova situação, onde alguns fatos que já existiam no contexto prático, através da disponibilidade tecnológica, começam a ser amparados pelo meio jurídico.

Atualmente, existem três projetos de lei no Congresso Nacional que abordam o tema assinatura digital. A seguir, serão apresentados um resumo destes projetos, com o relato de alguns autores, e no final desta dissertação (Anexo C), os três projetos se encontram na integra.

- **Projeto de Lei do Senado N.º 672 de 1999**

Este projeto segue o modelo de lei da UNCITRAL, de 1996, que procura regulamentar o comércio eletrônico de forma geral. Assim, pouco contribui para a assinatura digital especificamente. A abordagem deste projeto não estipula detalhadamente a tecnologia a ser utilizada. O fator principal desconsiderado pelo legislador refere-se ao fato de que existe, no meio da tecnologia de informação, uma série de elementos externos que podem vir a influenciar no conteúdo de um documento, e que a legislação não pode deixar ao encargo das partes a obrigação

de possuírem conhecimentos técnicos suficiente para estipularem o mecanismo que irá autenticar a operação.

- **Projeto de Lei da Câmara N.º 1.483 de 1999**

Este projeto foi criado visando instituir a fatura eletrônica e a assinatura digital nas transações de comércio eletrônico. No que diz respeito à assinatura digital, visa objetivamente à validação do mecanismo para as transações comerciais eletrônicas. Quanto à forma de reconhecimento da assinatura, propõe que a mesma seja feita por órgão público, sem, contudo especificar qual e como será o procedimento a ser adotado para a operacionalização do feito. O projeto em questão apresenta um aspecto excessivamente abstrato, buscando meramente regulamentar a existência da assinatura digital no país e remetendo a responsabilidade de estipular normas de funcionamento e controle para outros setores do Estado. No entanto, pode ser considerado de grande relevância, por ter iniciado um processo de mobilização do Poder Legislativo para a criação de uma regulamentação que ordene e controle a utilização da assinatura digital no Brasil (VOLPI, 2001).

- **Projeto de Lei da Câmara N.° 1589 de 1999**

Este projeto de Lei foi elaborado a partir do anteprojeto de lei desenvolvido pela comissão especial de informática jurídica da OAB-SP e dispõe sobre o comércio eletrônico, a validade jurídica do documento eletrônico e a assinatura digital. Trata-se de um projeto muito mais extenso e que busca atender um universo muito maior de situações que se remetam a esta matéria. Encontra-se nele inserido uma vasta regulamentação sobre o tratamento legal do comércio eletrônico, bem como dos documentos eletrônicos. Por essa razão será tratado com mais profundidade a seguir.

Logo em seu início, o projeto busca orientar para que a interpretação seja feita considerando-se o fato de que o comércio eletrônico opera inclusive de modo globalizado, e que seja levado em consideração o dinâmico processo de evolução que existe no meio tecnológico. Desta forma, busca favorecer a existência de uma sobrevida a esse tipo de comércio caso haja uma eventual alteração no mercado internacional ou mesmo no meio tecnológico.

A eficácia do projeto é direcionada objetivamente para o sistema assimétrico de assinatura, entendendo como original o documento que apresente assinatura mediante mecanismo criptográfico de chave pública.

Segundo o projeto, passa a ser presumido como verdadeiro o conteúdo do documento eletrônico, desde que apresente uma assinatura digital, seguindo todos os atuais princípios da tecnologia que envolve as assinaturas digitais. Assim, a assinatura digital deve:

- ser única e exclusiva para o documento assinado;
- ser passível de verificação;
- ser gerada sob o exclusivo controle do signatário;
- estar de tal modo ligada ao documento eletrônico que, em caso de posterior alteração deste, a assinatura seja invalidada;
- não ter sido gerada após a expiração, revogação ou suspensão das chaves;
- considerar como data do documento eletrônico a data em que foi registrado, a data da sua apresentação em repartição pública ou em juízo, ou, ainda, a data do ato ou fato que estabeleça, de modo certo, a anterioridade da formação do documento e respectivas assinaturas.

A figura do tabelião está presente no projeto que o responsabiliza pela emissão dos certificados digitais públicos. Assim como os Tabeliões convencionais, a atuação desses Tabeliões, no que diz respeito às assinaturas digitais e aos documentos eletrônicos, está ligada às mais diversas situações, inclusive fé perante terceiros. Desta forma é dotado de fé pública, possuindo, em sua essência, a função de garantir a publicidade, autenticidade, segurança e eficácia dos atos jurídicos.

Volpi (2001, p.55) relata que:

> Ao buscar junto ao tabelião a tarefa de garantir a autenticidade dos documentos eletrônicos perante terceiros, o projeto promove a manutenção do serviço notarial, dotado de fé pública e relacionado diretamente com o Estado. Este posicionamento figura de maneira interessante para a sociedade, uma vez que a responsabilidade da autoridade certificadora permanece diretamente vinculada ao poder público e não sob domínio exclusivo de uma entidade particular qualquer.

Segundo Lucca e Simão (2000, p.407), o projeto de lei estabelece as informações mínimas que o certificado digital deve conter e que seriam:

- identificação e assinatura digital do tabelião;
- data de emissão do certificado;
- identificação da chave pública e do seu titular, caso o certificado não seja diretamente apensado àquela;
- elementos que permitam identificar o sistema de cifragem utilizado;
- nome do titular e poder de representação de quem solicitou a certificação, no caso de o titular ser pessoa jurídica;
- a inclusão do prazo de validade do certificado;
- o número de série do certificado.

No projeto em questão fica clara a grande responsabilidade que cabe ao Ministério da Ciência e Tecnologia, levando-se em consideração que o mesmo emitirá certificados para chaves de assinatura a serem utilizadas pelos tabeliães para formarem os certificados digitais. Desta forma, este serviço exige extrema especialização técnica, devido ao ritmo dinâmico de atualização dos aspectos de segurança.

Lucca e Simão (2000, p.415) declaram que:

> Para regular o setor, poderia ser criada uma agência federal de certificação, a ser constituída, por exemplo, em conjunto pelo Poder Judiciário e pelo Ministério da Ciência e Tecnologia, com poder de polícia, função de analisar os pedidos de autorização para funcionamento, estabelecer regras e parâmetros para o exercício dessa atividade, sobre a tecnologia empregada etc. dever.

O tabelião, segundo aquele projeto de lei, irá revogar ou suspender o certificado digital quando:

- o titular da chave de assinatura ou de seu representante solicitarem;
- de ofício ou por determinação do Poder Judiciário, caso seja verificado que o certificado foi expedido baseado em informações falsas;
- se tiver encerrado suas atividades, sem que tenha sido sucedido por outro tabelião;

- houver ocorrência de dúvida sobre a legitimidade do requerente ou no que diz respeito à segurança da chave privada do titular.
- No que diz respeito às responsabilidades dos tabeliões, o projeto de lei estabelece, dentre outras, as seguintes atribuições:
- Publicar os certificados digitais, a fim de facilitar a verificação por parte dos usuários que necessitarem desta informação, que deve ser em tempo real e mediante acesso eletrônico remoto;
- Permitir o acesso do público a todas as chaves por ele certificadas.

Ainda no que diz respeito à certificação digital, Volpi (2001, p.59) relata que:

> Quanto ao tratamento da certificação digital, ressalta-se que o assunto é relacionado a um meio intangível, onde a confiança é depositada sobre informações que foram geradas a partir da computação de dados. Sob este aspecto, o projeto apresenta um ponto forte quando trata da fácil possibilidade da revogação ou suspensão da validade de um certificado digital.

Caso o tabelião venha a encerrar suas atividades, deverá assegurar que os certificados que tenha emitido sejam transferidos para outro tabelião, e, na eventualidade de não haver sucesso na transferência, cabe ao tabelião revogar os referidos certificados. Deverá ele, ainda, enviar ao Poder Judiciário as fichas de pedido de certificação que foram preenchidas pelos requerentes na época da inscrição e as demais documentações inerentes às fichas.

O não cumprimento dessas responsabilidades acarretará penalidades que podem ser impostas aos tabeliães, na mesma forma daquela que aborda as sanções penais relacionadas a fatos sujeitos à sanção de outros crimes já tipificados pelo Código Penal.

Quando se refere à possibilidade de falsificação do documento eletrônico, o projeto determina que o documento eletrônico é falso quando:

- assinado com chaves fraudulentamente geradas em nome de outrem;
- acontecer de o documento ser alterado sem que a chave original tenha sido modificada; nesse caso, será solicitada ajuda técnica, utilizando para tanto, um algoritmo de codificação de maior vulnerabilidade, para que se prove a autenticidade do documento.

Devido à importância da matéria do Projeto de Lei 1589, de 1999, que vai afetar a vida de milhões de brasileiros, Lucca e Simão (2000, p.415) afirmam que o mesmo:

> deveria estar melhor apresentado para que a sociedade civil organizada pudesse opinar, formular críticas e sugestões, a fim de aprimorar as propostas nele albergadas, adequando-o aos interesses e às necessidades dos setores público e privado, bem como da população em geral.

Todos os projetos citados anteriormente podem não refletir a regulamentação final sobre o assunto, entretanto, fornecem um forte direcionamento sobre como o tema será abordado na legislação do Brasil. (VOLPI,2001).

Em suma, o comércio eletrônico vem-se expandindo mundialmente. No Brasil, porém, ainda não há garantias suficientes, por não ter uma regulamentação forte sobre o assunto.

A legislação de outros países já trata sobre a validade do documento eletrônico, sobre a assinatura digital e sua certificação e sobre as normas aplicáveis ao comércio eletrônico. O quadro 2 a seguir, faz uma análise comparativa resumindo alguns aspectos legais tratados com maior freqüência em outros paises.

Quadro 2 – Análise comparativa da legislação adotada em outros países e por organismos internacionais

País	**Portugal**	**República Tcheca**	**Irlanda**	**Peru**
Instrumento legal	Decreto-Lei nº 290-D/99	Ato nº 227, de 29/6/2000 (The Electronic Signature Act)	Electronic Commerce Act, 2000	Ley nº 27269
A legislação inclui definições dos principais termos usados	SIM	SIM	SIM	NÃO (Remete para o regulamento)
Trata da validade do documento eletrônico	SIM	SIM (mensagem eletrônica)	SIM	NÂO
Trata da assinatura eletrônica	SIM	SIM	SIM	SIM
É neutra tecnologicamente	NÃO (criptografia assimétrica)	SIM	SIM	NÃO (criptografia assimétrica)
Trata da certificação	SIM	SIM	SIM	SIM
Admite o credenciamento da entidade certificadora	SIM (voluntário)	SIM (voluntário) (Administração pública só aceita documento eletrônico certificado por entidade credenciada	SIM (voluntário)	SIM (voluntário) (compulsório o registro)
Trata de certificadoras de outro país	SIM	SIM	NÃO	SIM
Trata da proteção à privacidade	SIM (somente de informações prestadas às entidades certificadoras)	NÃO	NÃO	SIM (somente de informações prestadas às entidades certificadoras)
Trata da proteção ao consumidor	NÃO	SIM (remete à legislação específica)	SIM	NÃO
Trata dos deveres e resposabilidades dos intermediários (provedores)	NÃO	NÃO	NÃO	NÃO
Inclui disposições tributárias	NÃO	NÃO	NÃO	NÃO

Quadro 2 – Análise comparativa da legislação adotada em outros países e por organismos internacionais (cont.)

País	**Colômbia**	**Espanha**	**Alemanha**	**Hong-Kong**
Instrumento legal	Ley 527 de 1999	Real Decreto-ley 14/1999	Law Governing Framework Conditions for Electronic Signatures	Electronic Transactions Ordinance
A legislação inclui definições dos principais termos usados	SIM	SIM	SIM	SIM
Trata da validade do documento eletrônico	SIM (mensagem eletrônica)	NÃO	NÃO	SIM
Trata da assinatura eletrônica	SIM	SIM	SIM	SIM
É neutra tecnologicamente	NÃO (criptografia assimétrica)	SIM	NÃO (criptografia assimétrica)	NÃO (criptografia assimétrica)
Trata da certificação	SIM	SIM	SIM	SIM
Admite o credenciamento da entidade certificadora	SIM (compulsória)	SIM (voluntária)	SIM (voluntária)	SIM (voluntária)
Trata de certificadoras de outro país	SIM	SIM	SIM	SIM
Trata da proteção à privacidade	NÃO	NÃO	SIM (somente de informações prestadas às entidades certificadoras)	SIM (somente de informações prestadas às entidades certificadoras)
Trata da proteção ao consumidor	NÃO	NÃO	NÃO	NÃO
Trata dos deveres e resposabilidades dos intermediários (provedores)	NÃO	NÃO	NÃO	NÃO
Inclui disposições tributárias	NÃO	NÃO	NÃO	NÃO

Quadro 2 – Análise comparativa da legislação adotada em outros países e por organismos internacionais (cont.)

País	**Cingapura**	**Estados Unidos**	**Comunidade Européia**	**UNCITRAL**
Instrumento legal	Electronic Transactions Act	Electronic Signatures in Global and National Commmerce Act	Diretiva 99/93-CE	Lei Modelo
A legislação inclui definições dos principais termos usados	SIM	NÃO	SIM	SIM
Trata da validade do documento eletrônico	SIM	SIM (mensagem eletrônica)	SIM	SIM (mensagem eletrônica)
Trata da assinatura eletrônica	SIM	SIM	SIM	SIM
É neutra tecnologicamente	NÃO (criptografia assimétrica)	SIM	SIM	SIM
Trata da certificação	SIM	NÃO	SIM	NÃO
Admite o credenciamento da entidade certificadora	SIM (voluntário)	NÃO	SIM (voluntário)	NÃO
Trata de certificadoras de outro país	SIM	NÃO	SIM	NÃO
Trata da proteção à privacidade	NÃO	NÃO	Remete a outra diretiva (95/46 – CE)	NÃO
Trata da proteção ao consumidor	NÃO	SIM (preserva direitos de outras legislações)	NÃO	NÃO
Trata dos deveres e resposabilidades dos intermediários (provedores)	SIM	NÃO	NÃO	NÃO
Inclui disposições tributárias	NÃO	NÃO	NÃO	NÃO

Fonte: CBEJI – Centro Brasileiro de Estudos Jurídicos da Internet

3 MODELAGEM DA ASSINATURA DIGITAL EM JAVA

Apesar das comercializações eletrônicas existirem bem antes do advento da *Internet*, foi através do seu surgimento que houve um gigantesco crescimento neste tipo de comercialização, surgindo assim à necessidade de segurança e integridade nos dados trocados entre as Instituições públicas ou privadas, e demais setores da sociedade moderna que desejam trocar informações por um meio digital, sendo este meio a *Internet* ou até mesmo uma rede privada de dados. Dentro deste quadro, a criação da autenticação eletrônica agregando as facilidades das assinaturas digitais e dos serviços a elas associados, permitindo a autenticação informatizada dos dados tornou-se um imperativo, pois as assinaturas digitais possibilitam ao destinatário dos dados eletronicamente enviados a verificação de autenticidade da origem e a integridade destes dados.

O objetivo deste capítulo é especificar uma modelagem da aplicação da assinatura digital em arquivos que necessitam ser assinados digitalmente, utilizando para esta modelagem a linguagem de programação Java. Esta linguagem foi propositalmente escolhida para esta modelagem, pois preenche a maioria dos quesitos necessários relacionados à questão de segurança, como veremos a seguir.

3.1 A Linguagem Java

Java é uma linguagem computacional completa, de alto nível de programação, orientada a objeto, projetada para ser portável entre diferentes plataformas e sistemas operacionais, possibilitando que o mesmo programa escrito nesta linguagem seja executado em computadores com sistemas operacionais diferenciados ou até mesmo em celular, *pager*, cartão de credito, *smart card*, etc. Bastando apenas que o equipamento em questão suporte uma Máquina Virtual Java (*Java Virtua Machine* - JVM). (CHAN, GRIFFITH, IASI, 1999)

3.1.1 Um breve histórico

Segundo a Sun Microsystem, os estudos e pesquisas sobre a linguagem Java tiveram inicio em 1991, por um pequeno grupo de engenheiros da própria Sun denominado *Green Project* que pretendia criar uma nova geração de computadores portáteis inteligentes, capazes de se comunicar de muitas formas, ampliando suas potencialidades de uso. Decidiram também criar uma nova plataforma para o desenvolvimento destes equipamentos de forma que seu *software* pudesse ser portado para os mais diferentes tipos de equipamentos. Para tal desenvolvimento, inicialmente escolheram a linguagem C++, aproveitando suas características e a experiência dos integrantes do grupo no desenvolvimento desta linguagem. Porém, nem mesmo o C++ permitia realizar com facilidade tudo aquilo que o grupo desejava. Diante destas dificuldades tecnológicas e a falta de uma linguagem que pudesse suprir as necessidades do projeto, o Coordenador James Gosling, decidiu então pela criação de uma nova linguagem de programação que pudesse conter tudo aquilo que era considerado importante e que ainda assim fosse simples, portátil e fácil de programar. Surgiu assim a linguagem interpretada chamada Oak (carvalho em inglês) que ganhou este nome devido ao fato de existir uma arvore destas em frente ao escritório de Gosling.

Após alguns anos de trabalho no desenvolvimento e aperfeiçoamento desta linguagem, surge a primeira grande oportunidade de aplicação. O desenvolvimento de uma tecnologia para TV a cabo interativa, porém como se tratava de uma licitação pública a Sun Microsystems acabou sendo vencida pela SGI (*Silicon Graphics Inc*.) fazendo com que a Oak ficasse sem uso definido até 1994.

Por problemas de *copyright*, o Oak, teve que ser rebatizado, recebendo desta forma o nome de Java, e após 1994, quando estimulados pelo grande crescimento da *Internet*, Jonathan Payne e Patrick Naughton desenvolveram o programa navegador WebRunner (apresentado formalmente pela Sun Microsystems como HotJava), capaz de efetuar o *download* e a execução de código Java via *Internet*.

No dia 23 de maio de 1995, a Sun Microsystems apresenta formalmente o Java para o mundo, poucos meses depois a Netscape Corp. lança uma nova versão

de seu navegador também capaz de efetuar o *download* e a execução de pequenas aplicações Java então chamadas de *applets*.

Em uma iniciativa também inédita a Sun Microsystems decide disponibilizar o Java gratuitamente para a comunidade de desenvolvimento de *software*, embora detenha todos os direitos relativos à linguagem e as ferramentas de sua autoria. Surge assim o Java Developer's Kit 1.0 (JDK 1.0). As plataformas inicialmente atendidas foram: Sun Solaris e Microsoft Windows 95/NT. Progressivamente foram disponibilizados kits para outras plataformas tais como Linux, Apple Macintosh, Novel, AIX, HPUX, e outros. (JANDL, 2002).

3.1.2 Princípios básicos e principais características da linguagem

A linguagem Java possui importantes características que diferenciam das demais linguagem de programação existente hoje no mercado. Entre as principais características que tornaram essa linguagem tão eficiente, destaca-se a portabilidade, orientação a objetos, Multitarefa (*multithreading*), segurança e familiaridade com as linguagens C e C++.

- **Portabilidade**

A independência de plataforma é uma das principais vantagens do Java em relações a outras linguagens de programação, principalmente quando se deseja que um programa seja executado em vários tipos de plataformas diferentes.

Quando se escreve um programa em uma linguagem como o C, C++, Pascal, Visual Basic, etc, os compiladores ou interpretadores destas linguagens geram um programa executável de baixo nível, popularmente conhecido como linguagem de máquina. Entretanto, este programa executável que foi gerado em um sistema operacional especifico, e em uma plataforma de processador especifico, só poderá ser executado em computadores com as mesmas características, sendo assim, um programa for escrito em C++, e gerado o seu executável em um computador cujo o sistema operacional seja Linux com arquitetura de Processador

Intel ou compatível, este programa só poderá ser executado em outro computador com sistema operacional Linux e com a mesma arquitetura de processador. Caso haja a necessidade deste programa ser executado em outro tipo de plataforma, é necessário obter um compilador em C++ para a plataforma desejada e re-compilar o programa, e em alguns casos realizar algumas alterações no código do programa fonte.

A situação é diferente quando se escreve um código de programação em Java. Programadores Java, em todo o mundo costuma dizer a seguinte frase: “Escreva uma vez, execute em qualquer lugar”. (OAKS, 1999).

O ambiente de desenvolvimento do Java é composto por duas partes, um compilador e um interpretador. O compilador compila o código de programação em um programa intermediário chamado de Java *bytecode* e armazenado em um arquivo com extensão “.class” destinado a uma única plataforma, a Maquina Virtual Java que seria o interpretador do *bytecode*. A JVM nada mais é do que um computador dentro de outro (computador virtual interno), possibilitado que um *bytecode* seja executado em qualquer computador, bastando apenas que este computador tenha uma JVM instalada (JANDL, 2002). Nos dias de hoje, já existe JVM para vários tipos de plataformas, inclusive celulares, pagers, televisores, etc.

- **Orientado a Objetos**

Java é uma linguagem de programação Orientada a Objetos, pois com exceção dos seus tipos primitivos de dados, tudo em Java são classes ou instância de uma classe.(JANDL, 2002).

Uma das diferenças básica entre a programação estruturada convencional e a programação orientada a objetos é algo chamado encapsulamento. O encapsulamento permite controlar o acesso aos atributos e aos métodos que agem dentro de um objeto, é uma maneira de fazer com que um objeto exponha apenas o que é necessário ser exposto, mantendo como privado os dados (atributos) dentro de uma função (métodos).

- **Multitarefa**

O Java oferece recursos para desenvolver programas capaz de executar várias tarefas compartilhando o tempo do processador. Cada um destes fluxos de execução é o que se denomina *Thread*, tornando um importante recurso de programação para aplicativos mais sofisticados. (JANDL, 2002)

- **Semelhança com C e C++**

O Java é derivado da Linguagem C, podendo ser comparado com um C++, porém mais simplificado. Esta semelhança não deve ser mera coincidência, pode ter sido escolhida propositalmente pela grande quantidades de programadores em C/C++ existente no mercado, desta forma a linguagem Java já nasceria com uma quantidade elevada de profissionais capacitados, ou pelo menos semicapacitados para o desenvolvimento de *softwares* utilizando esta linguagem.

Ao contrario do C/C++, o Java não possui ponteiros (endereço de memória, onde o valor indica “onde” uma variável está armazenada e não “o que” esta armazenado), isso significa que não permite a manipulação direta de endereços na memória, nem exige que os objetos criados sejam destruídos. Além disso, a JVM possui um mecanismo que gerencia automaticamente a memória, recuperando a memória alocada para objetos não mais referenciados pelo programa, liberando a RAM (*Random Access Memory)* para outros processos. Este dispositivo é conhecido pelo nome de *Garbage Collector*. (JANDL, 2002).

Para aqueles programadores familiarizados com C/C++, observa que a maior semelhança com o Java é a sintaxe de programação, mais além da inexistência de ponteiros como citado anteriormente, o Java apresenta outras diferenças como, por exemplo: Não possuir a instrução *goto;* Permite passar parâmetros do tipo A*rray* e *Objeto* por referência; etc. Porém estas diferenças não serão detalhadas por não ser o foco deste capítulo.

- **Segurança**

Nos dias atuais, pelo fato do Java estar fortemente associado à *Internet*, um dos aspectos mais divulgados da linguagem é sua segurança. Segundo Schneier

(200, p.170), “a linguagem Java é a única linguagem de programação projetada especialmente para o código móvel, e com a segurança em mente”. Por isso oferece várias camadas de controle de segurança que protegem contra códigos maliciosos, permitindo que os usuários rodem tranqüilamente programas de origem desconhecida, como os *applets* – que são programas executados por um navegador de paginas da internet (*browser).* (KNUDSEN, 1998).

Outra camada de proteção para segurança é comumente chamada de modelo de caixa de areia *sandbox*, onde, por exemplo, um aplicativo *applet* Java de origem desconhecida pode se executado, porém é mantido isolado dentro desta caixa de areia, podendo ser executado com segurança evitando danos ao ambiente que esta sendo executado.

Segundo Schneier (2001, p.170) a caixa de areia é protegida por três mecanismos:

Verificador de código de bytes: sempre que um *browser* faz um *download* de um *applet* Java, o verificador de código analisa o código primeiro. O verificador garante que o código de *bytes* esteja formatado corretamente e não possua qualquer um dos vários problemas comuns.

Carregador de classe: esse componente determina como e quando um *applet* pode ser incluído no ambiente Java, certificando-se de que o *applet* não substituirá algo importante que já exista.

Gerenciador de segurança*:* é como um monitor de referência. Ele é consultado sempre que o *applet* Java tenta algo questionável como abrir ou gravar um arquivo ou abrir uma conexão na rede, por exemplo. Dependendo de como o *applet* foi instalado, essas permissões serão permitidas ou negadas.

Para Oaks (1999, p. 02), o termo “segurança” tem suas restrições no uso, entretanto as diferentes expectativas podem encaminhar para os seguintes itens quando se trata de programas escritos na linguagem Java:

- **Seguros contra programas maliciosos:** Programas não devem ter permissão para danificar o ambiente computacional de um usuário, incluindo cavalo de tróia (*trojans),* bem com programas nocivos que podem duplicar-se (vírus de computador).
- **Não-intrusos:** Deve-se evitar que os programas descubram informações privadas do computador *host* ou na rede de um computador *host*.

- **Autenticados:** A identidade das partes envolvidas no programa deve ser verificada.
- **Criptografados:** Os dados que o programa envia e recebe devem ser criptografados.
- **Examinados:** As operações potencialmente sensíveis devem sempre ser conectadas.
- **Bem-definidos:** Uma especificação de segurança bem-definida deve ser seguida.
- **Verificados:** As regras de operação devem ser definidas e verificadas.
- **Bem-comportados:** Deve-se evitar que os programas consumam muitos recursos do sistema.

3.1.3 O Pacote de Segurança Java

O Pacote de Segurança da linguagem Java (*package Java security*) é uma *Apllication Programming Interface* (API) complexa que permite a inclusão de dispositivos de segurança genéricos em suas applets e aplicações. Neste pacote, existe um amplo conjunto de classes, dentre tantas funcionalidades, esta API pode ser usada para gerar criptografia, autenticação, assinatura digital, chaves secretas e pares de chaves (pública e privada).

As implementações concretas destas classes são fornecidas pela empresa Sun Microsystem no *Java Development Kit* – JDK, e só se tornaram possíveis através da infra-estrutura do provedor de segurança.

Segundo Oaks (1999, p.171):

> Os provedores de segurança são a união que gerencia o mapeamento entre os motores usados pelo restante do pacote de segurança (como a compilação de mensagem), os algoritmos específicos que são válidos para tais motores (como uma compilação SHA) e as implementações específicas deste par algoritmo motor que pode estar disponível para qualquer máquina Java em particular.

No JDK versão 1.1, um *provider* poderia conter uma implementação de um ou mais algoritmos de assinatura digital, já no Java 2, aparecem cinco tipos adicionais de serviço, ou seja, além de algoritmos de *hash*, e de geração de chaves, o JDK passou a conter em seu pacote de segurança fábricas de chaves, criação e gestão de *keystores* (arquivos que servem como um banco de dados de certificados e chaves privadas, garantindo uma centralização dos certificados e chaves), geração e gestão de parâmetros de algoritmos e de fábricas de certificados, e ainda a possibilidade de implementar algoritmos de geração de números pseudo-aleatórios (BARBOSA, 2001). A descrição destas implementações será apresentada no decorrer deste capítulo.

Os provedores de segurança contam com a cooperação entre si e os demais pacotes de segurança Java para alcançar seu objetivo. Assim, torna-se responsabilidade da classe que está sendo usada perguntar à classe de segurança (*Security class)* qual classe em particular será usada para executar determinada tarefa. A classe de segurança, por sua vez, pergunta a cada um dos provedores se pode ou não fornecer a compilação desejada. Portanto, um programa típico que deseja usar o pacote de segurança não interage diretamente com o provedor de segurança.

Para realizar uma determinada operação, o pacote de segurança construirá uma *string*, representando-a, e pedirá à classe de segurança um objeto que possa realizar a operação com o algoritmo dado. É importante salientar que nem todo algoritmo pode ser usado para realizar toda operação; as combinações válidas são listadas no quadro 3. As entradas em *itálico* e <u>sublinhado</u> do quadro 3 são operações que a especificação de segurança do Java define como legais, mas não são implementadas pelo provedor de segurança padrão.

Quadro 3 – Recursos de segurança e algoritmos esperados na API de segurança.

Motor (operações)	Nome do Algoritmo
AlgorithmParameters *	DSA
AlgorithmParametersGenerator*	DSA
KeyFactory *	DSA
KeyPairGenerator	DSA
KeyPairGenerator	*RSA*
MessageDigest	MD5
MessageDigest	SHA-1
MessageDigest	*MD2*
Signature	DSA
Signature	*MD2/RSA*
Signature	*MD5/RSA*
Signature	*SHA-1/RSA*

Fonte: Oaks, Scott (1999, p.173)

3.1.3.1 Arquitetura de Criptografia Java (JCA)

O projeto global da criptografia de classes é gerenciado pela Arquitetura de Criptografia Java (*Java Cryptography Architecture* – JCA). O JCA é um *framework* que facilita a incorporação de funcionalidades criptográficas nas aplicações Java. Por ser um *framework*, a JCA possui um conjunto de classes que interagem entre si, com o intuito de disponibilizar uma solução total ou parcial para um problema, podendo criar aplicações que utilizam recursos oriundos de assinatura digital mais facilmente, sem exigir um grande conhecimento no assunto (BARBOSA, 2001).

Introduzindo o conceito de *Criptography Package Provider* – CPP, um ou vários pacotes que fornecem uma implementação concreta de um subconjunto dos serviços definidos na API, a JCA foi desenvolvido para separar conceitos de

criptografia e implementações. Os conceitos são encapsulados por classes nos pacotes *java.security* e *javax.crypto*. As implementações são fornecidas pelos provedores de criptografia (*cryptographic providers*) da JCA (KNUDSEN, 1998).

A JCA foi projetada para conter os seguintes objetivos:

- **Independência das implementações específicas de técnicas criptográficas:** permite que as aplicações possam usar um determinado algoritmo criptográfico sem se preocupar com as especificidades das suas várias implementações. Estas independências são adquiridas através do conceito de *providers,* e quando a aplicação requisita um determinado algoritmo ao *framework*, lhe é fornecida uma implementação de qualquer um dos *providers* instalados, permitindo que a mesma aplicação seja executada em plataformas com diferentes *providers* instalados, porém estes devem implementar o mesmo algoritmo.
- **Compatibilidade entre as implementações:** possibilita que as chaves geradas por um *provider* possam ser utilizadas por outro desde que utilizando o mesmo algoritmo, ou que as assinaturas geradas por um *provider* possam ser verificadas por outro.
- **Independência dos algoritmos:** permite a utilização de diversas técnicas criptográficas existentes, independentemente do algoritmo que as implementam. Isto se deve ao fato de classes *engine* (motor) abstratas que especificam cada uma das técnicas criptográficas.
- **Possibilidade de extensão:** possibilita que novos *providers* possam ser acrescentados ao *framework*, porém estes *providers* devem disponibilizar implementações das *engine class*.

A figura 15 ilustra como o JCA abrange as classes incluídas no JDK bem como os grupos de *software* API do Java *Security*.

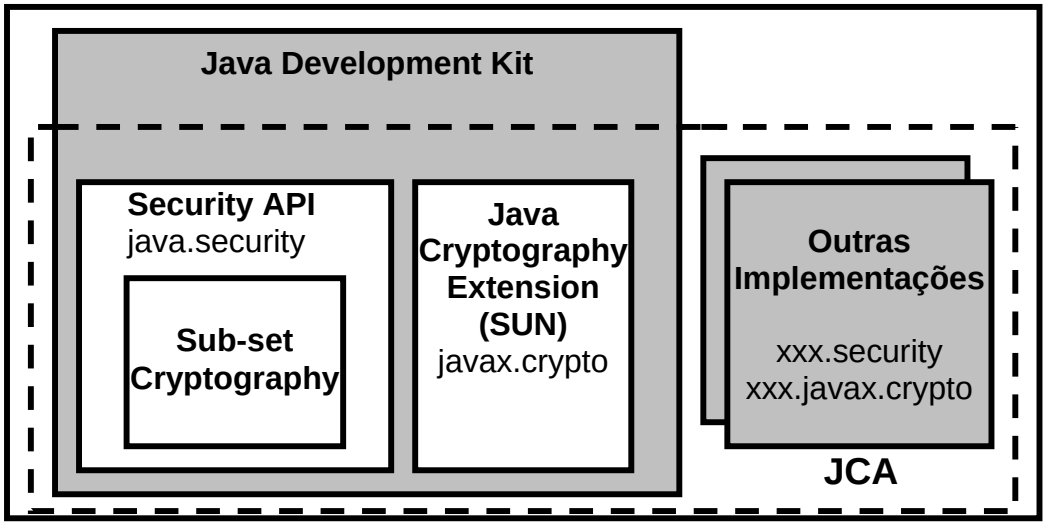

FIGURA 15 – Componente da JCA (Java Cryptography Architeture).

Das várias classes incluídas na JCA, serão listadas as mais importantes e que são base para a funcionalidade de uma assinatura digital.

- **Classe Engine**

Esta classe define um serviço criptográfico de forma abstrata, independente do algoritmo e da implementação, e define uma API para este serviço. Um serviço pode estar associado a operações como: efetuá-las quando estas forem criptográficas, como a assinatura digital ou o *hashing*; gerar material criptográfico (chaves ou parâmetros) necessário para efetuar operações criptográficas; gerar objetos de dados (repositórios de chaves - *keystores* ou certificados) que encapsulam chaves criptográficas de uma forma segura, dentre outras (ARNOLD E GOSLING, 1997).

- **Classe MessageDigest**

Esta é a classe de compilações de mensagens. Além de serem componentes das assinaturas digitais, também são úteis para verificar se um conjunto de dados não foi corrompido. Através de seu uso é possível gerar um valor de prova para qualquer entrada arbitrária O valor da prova é indiferentemente chamado de impressão digital ou compilação. Estas compilações possuem duas

prioridades, podendo gerar uma prova única para um conjunto de dados de entrada. Os dados de entrada originais são indiscerníveis, a partir da saída da prova. (OAKS, 1999).

- **Classe SecureRandon**

Esta classe é dedicada às operações de geração de números aleatórios ou pseudo-aleatórios (PRNG).

Segundo Oaks (1999, p.382), “ao contrário do gerador de números aleatórios padrão, os números gerados por esta classe são criptograficamente seguros”, ou seja, estão menos sujeitos à adivinhação do padrão e a outros ataques que podem ser feitos em um gerador de números aleatórios tradicional.

- **Classe KeyPairGenerator**

A classe KeyPairGenerator é um dos *engines* padrão, podendo ser fornecido por um provedor de segurança Java, que irá gerar uma chave pública, bem como sua chave privada relacionada. As instâncias desta classe irão gerar pares de chaves apropriados para um algoritmo particular (DAS, RSA, etc). Implementado pela maioria dos geradores de chave, quando for inicializado, retorna às chaves de uma resistência em particular, normalmente o número de bits da chave, mas também pode ser inicializado de uma forma especifica do algoritmo.(OAKS, 1999).

- **Classe Signature**

A classe Signature é a Engine Class dedicada à operações criptográficas de assinatura digital e verificações destas assinaturas através de diferentes tipos de algoritmos. Numa assinatura digital, uma chave pública e um documento (mensagem) com comprimento arbitrário são utilizados para gerar uma assinatura, atestando a autenticidade da mensagem e permitindo a integridade dos dados. O objeto da classe Signature deve ser inicializado com a chave privada, quando usada para assinar, ou chave pública apropriada, quando utilizada para verificar a mensagem. Desta forma, a assinatura poderá ser gerada a partir dos dados inseridos em um arquivo e depois verificada (BARBOSA, 2001).

Segundo Barbosa (2001), Os Algoritmos de assinatura digital mais comum são:

- DSA com SHA-1 (SHA1withDSA);
- RSA com MD2 (MD2withRSA);
- RSA com MD5 (MD5withRSA);
- RSA com SHA-1 (SHA1withRSA).

Apesar de sua semelhança com a classe MessageDigest – pois, para um desenvolvedor gerar uma assinatura digital é realmente o mesmo que gerar a compilação de uma mensagem – a única diferença entre elas é o fato de que, na classe Signature, uma chave terá que ser apresentada para funcionar em objeto de assinatura. Esta diferença é bastante importante, pois uma compilação de mensagem assinada não poderá ser alterada sem o conhecimento da chave que foi usada para criá-la. O mesmo não acontece em uma compilação de mensagem em que a mesma pode ser alterada junto com os dados que ela representa, de tal forma que a adulteração seja imperceptível. (OAKS, 1999).

3.1.3.2 Extensão da Criptografia Java (JCE)

Criado separadamente devido às restrições de exportação de tecnologia criptográfica dos Estados Unidos e Canadá, a JCE (*Java Cryptography Extension*) baseia-se nos mesmos princípios da JCA: independência da implementação e, sempre que possível, a independência do algoritmo, utilizando a mesma arquitetura de *provider*. Era previamente um pacote opcional, que não podia ser obtido separadamente do JDK padrão (Java 2 SDK), versões 1.2.x e 1.3.x. No entanto, a JCE tem sido integrada na versão atual (Java 2 SDK, v 1.4.x)

Segundo Oaks (1999, p.278), ”A JCE segue a mesma infra-estrutura do provedor de segurança do restante da arquitetura de segurança de Java, é composta por um provedor de segurança adicional que inclui as implementações dos motores da JCE”.

Segundo Barbosa (2001), todas as tecnologias abrangidas pela restrição de exportação tecnológica foram implementadas apenas neste pacote de extensão, sendo elas as seguintes tecnologias:

- Algoritmo de cifra simétrica como o RC4;
- Algoritmo de cifra simétrica por bloco como o DES, RC2 e o IDEA;
- Algoritmo de cifra assimétrica como o RSA;
- Algoritmo de cifra baseado em *password* (senha) como o PBE. Este tipo de algoritmo gera a chave com base em uma *password* (senha). Para tornar o processo mais seguro, a maior parte dos algoritmos deste tipo introduz fatores aleatórios no processo de geração da chave.
- MAC (*Message Authentication Codes*), permite verificar a integridade de informação transmitida ou armazenada em um meio não confiável. Normalmente são utilizados entre duas partes que partilham uma chave secreta para validar as informações trocadas entre si.
- Key Agreement, este protocolo possibilita que duas ou mais entidades estabeleçam as mesmas chaves criptográficas sem trocarem informações secretas.

Assim como a JCA, a JCE também possui várias classes, das quais serão listadas as que mais se destacam neste pacote de extensão.

- **Classe Cipher**

Esta classe constitui o núcleo da JCE, fornecendo algoritmos criptográficos para cifrar e decifrar os dados. Tendo como formato algoritmo / modo e preenchimento (*algorithm/mode/padding),* ou seja, além de especificar um algoritmo criptográfico, pode se especificar um modo de funcionamento e um esquema de preenchimento, sendo que os dois últimos não devem ser especificados. Serão utilizados os valores padrão (default) da classe. Estes dados dependem do provedor de segurança utilizado, no caso do provedor da Sun Microsystem SunJCE, os modos de funcionamento principais são:

- ECB (*Eletronic Code Book*): muitas vezes utilizados para cifrar chaves, este modo cifra blocos de dados em blocos cifrados de forma

independente. Caso não seja especificado nada na utilização da classe *Cipher*, este modo atua como padrão.

- CBC (*Cipher Block chaining*): introduz segurança adicional na forma de um mecanismo de feedback entre blocos. Sendo assim, a decifragem de um bloco só é possível se o anterior for conhecido na sua versão cifrada. Este modo é muito utilizado para a cifragem de arquivos.
- PCBC (*Propagated Cipher Block Chaining*): variação do modo CBC em que o *feedbak* é estendido para incluir também a versão decifrada do bloco anterior.
- CFB (*Cipher Feedback*): este é o modo de retorno do criptograma, semelhante ao CBC, porém sua implementação interna é um pouco diferenciada, sendo que o CBC requer um bloco de dados de 8 *bytes* para começar sua codificação, enquanto o CFB pode começar com uma quantidade menor de dados.
- OFB (*Output Feedback*): este é o modo de retorno da saída, assim como o modo CFB utiliza algoritmo de cifra por blocos para gerar uma *stream keys* (fluxo de chaves). Entretanto os dados não estão implicados na geração das chaves, sendo que apenas a última chave serve como base para a geração da chave seguinte.

Tanto o modo CFB quanto o modo OFB permitem adicionar um número ao código do modo, com o tamanho das chaves (BARBOSA, 2001).

Os esquemas de preenchimento especificados pelo provedor de segurança SunJCE são:

- **PKCS5Padding**: assegura que os dados de entrada sejam preenchidos com um múltiplo de oito *bytes*.
- **NoPadding**: assegura que não seja feito nenhum preenchimento. Neste caso, o número de *bytes* de entrada terá que ser um múltiplo do tamanho do bloco do criptograma, senão quando o criptograma tentar codificar ou decodificar os dados, um erro será gerado (OAKS, 1999).

- **Classe SealedObject**

Esta classe permite criar um objeto e proteger a sua confidencialidade com um algoritmo criptográfico. Dado qualquer objeto que implemente a interface de serialização, é possível encapsular a forma serializada desse objeto dentro de um *SealedObject* que o "esconderá" em uma forma cifrada, sendo que o conteúdo protegido pode, posteriormente, ser decifrado e o processo de serialização poderá, então, ser invertido para obter o objeto original. A recuperação do objeto original pode ser feita de duas formas, sendo que a primeira utiliza um objeto do tipo *Cipher,* e a segunda, a chave secreta apropriada.

- **Classe MAC**

A enginie class MAC fornece a funcionalidade de um *Message Authentication Code* (MAC). A utilização de um objeto Mac é bem semelhante à do objeto *Cipher*, baseando-se nos métodos *update* e *doFinal,* permitindo este último obter o MAC pretendido.

Depois de obtida uma instância de um objeto deste tipo, ele deve ser inicializado utilizando o método *init,* que aceita como parâmetro a chave secreta a ser utilizada. Uma inicialização especifica de um algoritmo escolhido, com base em objetos do tipo *AlgorithmParameterSpec*, também pode ser utilizada.

3.2 A Linguagem Java na Implementação do Pacote de Assinatura Digital

Devido ao suporte dado pela API Java à criptografia e ao tratamento de dados necessários à implementação das tecnologias que envolvem as assinaturas digitais, é possível utilizar os recursos oferecidos pela linguagem através do pacote de segurança (*java.security*) para melhorar a confiabilidade e segurança dos dados trafegados em uma rede de computadores. Mesmo existindo algumas restrições à exportação de algumas ferramentas criptográficas, algoritmos de *hash* e chave

pública-privada estão disponíveis, possibilitando a criação de um protocolo local para assinar documentos digitalmente.

Utilizando o pacote de segurança do Java, juntamente com os componentes JCA (*Java Cryptography Architecture*) e JCE (*Java Cryptography Extension*), é possível criar um conjunto de classes que facilitará futuras programações baseadas em assinar um documento digitalmente. Este conjunto de classes, forma o pacote de assinatura digital, criando assim um componente computacional capaz de auxiliar o desenvolvimento de sistemas computacionais relacionados à implementação de uma assinatura digital, na qual todo o suporte para a criação deste pacote está disponível em versões da linguagem Java, a partir da versão 1.2 do JDK.

A figura 16, abaixo, demonstra um processo básico da assinatura digital. A primeira atitude que deve ser tomada, é a criação do par de chaves, sendo uma chave privada e uma chave pública. Após a criação do par de chaves, utiliza-se a chave privada gerada no processo anterior, e assina digitalmente o documento, gerando assim uma assinatura digital. Caso deseje verificar a validade da assinatura do documento, utiliza-se a chave pública gerada no primeiro processo; a assinatura digital do documeto gerada no segundo processo e o documento que deseja varificar a assinatura. Caso a chave pública for válida, e o documento assinado não sofreu nenhuma alteração, a assinatura digital é confirmada.

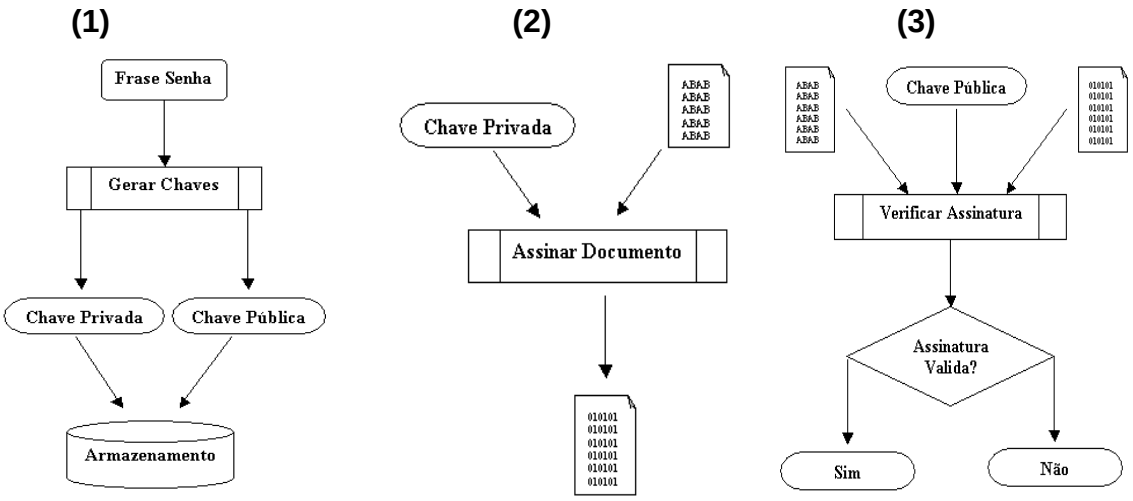

FIGURA 16 – Modelagem do processo básico da assinatura digital

Para todos os passos citados são requeridos algoritmos específicos, que são passíveis de utilização a partir da linguagem Java. O pacote em Java *(package*

javax.crypto) e seus sub-pacotes (*subpackages*) são os responsáveis pela criptografia forte em Java e, dentro desses pacotes, é possível encontrar praticamente todas as ferramentas necessárias para implementar uma assinatura digital.

Dentre os algoritmos analisados na presente pesquisa, o que se mostrou mais adequado para modelagem da assinatura digital que tem sido buscada, foi o algoritmo DSA, tanto para gerar as chaves quanto para efetuar e verificar a assinatura, por ser considerado seguro pelos criptógrafos, livres de patente e de fácil implementação em Java. Dentre outras características citadas anteriormente, o DSA oferece segurança em relação ao tamanho das chaves (1.024 bits), apresenta-se como um algoritmo padrão de assinatura, com facilidades de implementação em Java e principalmente é um algoritmo livre de patente, assim como o algoritmo de mensagem SHA-1.

Para se criar o pacote de assinatura digital, foram focados os três passos básicos para assinar um documento digitalmente, e acrescentados outros recursos que poderão ser úteis na hora da criação de um programa que os utilize. Os processos de geração das chaves, assinatura e verificação da assinatura serão melhor detalhados a seguir, descrevendo os procedimentos adotados em Java e os algoritmos utilizados na prática destes procedimentos.

- **Geração das chaves:**

A geração do par de chaves, é o primeiro processo para a realização da assinatura digital. O par de chaves gerada neste processo consiste em uma chave pública e uma chave privada, relacionadas matematicamente entre si. É de fundamental importância a geração do par de chaves, pois a chave privada é necessária para efetuar a assinatura digital, assim como a chave pública é necessária para verificar se a assinatura digital é válida ou não.

O processo de geração do par de chaves assimétricas, que é iniciada ao informar uma “frase-senha” ou “semente (*seed*)”, podendo ser uma seqüência tanto de números como de letras, ou a combinação de números e letras.

A figura 17 ilustra a geração do par de chaves utilizado no processo de assinatura digital.

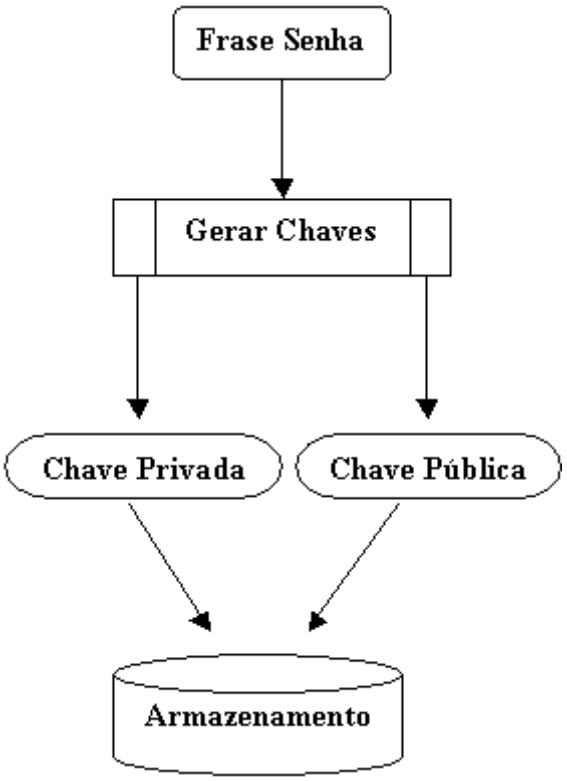

FIGURA 17 – Modelagem do processo de geração das chaves

Dada a “frase-senha”, será aplicado, sobre seus dados, um algoritmo de números pseudo-aleatórios, que efetivará o cálculo de um número a partir dos caracteres digitados, gerando uma série de números pseudo-aleatórios. A classe responsável por fornecer esses números pseudo-aleatórios é chamada de ***SecureRandom***. Ela gera números aleatórios que dificilmente se repetirão.

A geração das chaves é possível através da classe ***KeyPairGenerator*** e do algoritmo DSA para gerar as chaves. O tamanho da chave deve ser compatível com o algoritmo sendo que, no caso do DSA utilizado, o tamanho pode variar de 512 a 1.024 bits e tem que ser múltiplo de 64.(OAKS, 1999).

O algoritmo de mensagem SHA-1 aplicado à série de números pseudo-aleatórios obtido, gera um resumo, e o DSA aplicado a este resumo, irá gerar duas chaves: uma privada e uma pública, sendo aconselhável armazená-las em arquivos separados (por questão de segurança). No JCE do Java existem objetos adequados para o armazenamento dessas chaves. O armazenamento da chave privada deverá ser feito obrigatoriamente de forma a mantê-la protegida, dificultado a falsificação da assinatura digital.

- **Geração da assinatura digital:**

Após a geração do par de chaves, o processo de assinatura poderá ser realizado de acordo com o esquema ilustrado na figura 18, abaixo:

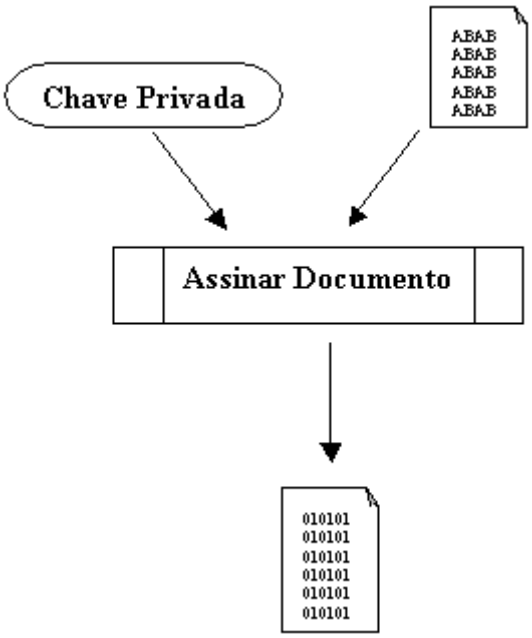

FIGURA 18 – Processo de assinatura

Tendo como base os algoritmos SHA1 e DSA, o processo ilustrado na figura 18 gera a assinatura de um arquivo. É importante ressaltar que tanto as chaves quanto a assinatura devem ser geradas utilizando o mesmo algoritmo de criptografia.

A classe ***Signature*** no Java é responsável por gerar as assinaturas digitais. Tendo em mãos a chave privada gerada no processo anterior, e um arquivo que se deseja assinar, já é possível iniciar o processo de assinatura digital propriamente dito, aplicando ao arquivo original o algoritmo de mensagem SHA-1 (que irá gerar um resumo) e o algoritmo de assinatura DSA (que iniciará a criação da assinatura digital). O método da classe *Signature* usado para gerar a assinatura em Java é chamado de ***sign.***

Após efetuar a assinatura, os dados gerados neste processo podem ser salvos junto com o resumo, que servirá como assinatura. Desta forma, tanto os dados quanto a assinatura poderão ser recuperados posteriormente. Se o arquivo assinado sofrer qualquer alteração, perderá automaticamente a assinatura digital, tendo que repetir o processo de assinatura.

Aqui termina a responsabilidade do assinante. Tudo o que ele precisa fazer agora é fornecer o arquivo assinado, juntamente com a chave pública e o arquivo em que está armazenado a assinatura digital correspondente.

- **Verificação da assinatura digital:**

Para validar a assinatura digital, é necessário possuir a chave pública, a assinatura digital e o arquivo assinado, que será verificado. A figura 19 ilustra de que forma é feita esta validação.

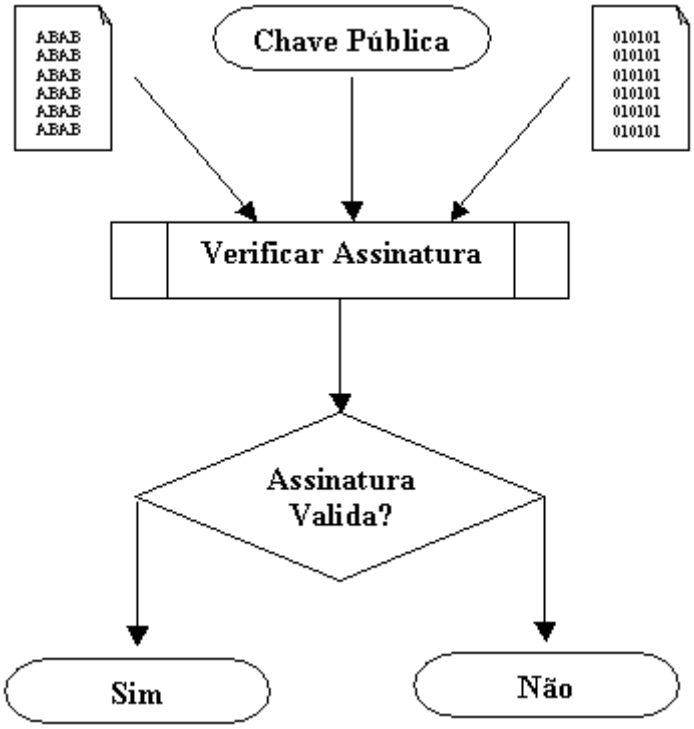

FIGURA 19 – Processo de verificação da assinatura

Iniciando o processo de verificação do arquivo assinado, após a abertura do arquivo original, aplica-se o algoritmo de mensagem SHA-1 no resumo (o qual foi enviado juntamente com o arquivo) e, para verificação da integridade da mensagem, utiliza-se a chave pública e o algoritmo DSA sobre a assinatura, sendo que ambos produzirão resumos que serão comparados, se as respostas dos resumos forem iguais, a assinatura será confirmada. Em Java, o método ***verify*** é responsável para efetuar a verificação da assinatura digital**.**

O conjunto de classes que formam o pacote de assinatura digital proposto contendo os principais recursos para efetuar a assinatura digital e a verificação do arquivo assinado, assim como a criação do par de chaves necessário e o código fonte em Java de cada classe deste pacote encontram-se em anexo no final desta dissertação (Anexo A).

4 APLICAÇÃO DOS RECURSOS DE ASSINATURA DIGITAL UTILIZANDO O COMPONENTE COMPUTACIONAL DESENVOLVIDO PARA O SISTEMA DE INFORMAÇÕES PROCESSUAIS

Nos dias atuais, em uma repartição pública ou em uma empresa privada, por menor que seja, circulam um enorme número de documentos, interna ou externamente. Estes documentos normalmente são impressos em papéis e armazenados em arquivos de aço, estando sujeitos a perdas parciais ou totais em virtude de má conservação, ações da natureza ou acidentes catastróficos. A idéia de se armazenar estes documentos em arquivos virtuais e guardá-los em mídias eletrônicas, diminuiria em muito estas perdas, podendo realizar inúmeras cópias de segurança, e serem restaurada quando necessário. Esta solução hoje em dia é muito comum, porém este processo teria uma falha de autenticidade, pois estes documentos guardados em mídias poderiam ser alterados por qualquer pessoa, impossibilitando saber se o seu conteúdo é autêntico ou não. Surge então, a idéia de utilizar um algoritmo de assinatura digital que valide estes documentos e garantam sua autenticidade.

O objetivo deste capítulo é apresentar um modelo computacional que assine um documento digitalmente, garantindo a autenticidade de processos e documentos em uma repartição pública, levando-se em consideração a parceria estabelecida entre o DesignLab (Laboratório de Design) da Universidade Federal de Santa Catarina, e a Justiça Trabalhista Brasileira, na sistematização do acompanhamento processual em todas às suas fases e etapas. Este acompanhamento será feito mediante a utilização das tecnologias que envolvem a assinatura digital e a modelagem de implementação proposta em Java.

O modelo proposto foi baseado no acompanhamento das funcionalidades do sistema atual, na disponibilidade de recursos físicos, na pesquisa das tecnologias inerentes à assinatura digital e na viabilidade de integração dessa técnica no aperfeiçoamento da segurança do Sistema de Informações Processuais.

4.1 Descrição do Sistema de Informações Processuais

O aplicativo atualmente em uso na Primeira Instância da Justiça Trabalhista é denominado Sistema de Informações Processuais (SIP), e tem por objetivo a realização de tarefas jurisdicionais, a melhoria das condições de trabalho dos serventuários da justiça e a agilidade dos serviços prestados à comunidade. O sistema tem por base uma filosofia de integração total, possibilitando, assim, que os dados sejam alimentados somente uma vez e, posteriormente, estejam disponíveis a todos os usuários, de acordo com critérios pré-estabelecidos.

O SIP é um aplicativo que tem como tarefa principal o acompanhamento processual em primeira instância, no Poder Judiciário, e também o auxílio para consultas a esses processos que podem envolver: cálculo de custas, controle estatístico, emissão de relatórios diversos, entre outros. Sua característica básica é a informatização de todas as fases da ritualística processual, que se inicia com o cadastramento de todos os dados correspondentes ao processo, e continua com o acompanhamento da distribuição, da autuação, do acompanhamento junto às Varas e Cartórios, do registro e controle de audiências, dentre outros. Todo esse processo fornece apoio para a elaboração de sentenças e despachos e facilidades para emissão e gerenciamento de documentos (certidões, mandados judiciais, despachos, editais, atas, outros expedientes e publicações legais, cálculos, registros e controles de guias de custas, além de possibilitar consultas diversas).

O Sistema é formado por uma base de dados única que serve para utilização nos processo de Segunda instância, havendo, portanto, um reaproveitamento de informações básicas e evitando a duplicidade das mesmas.

Dentre as principais características que o sistema propiciou à Justiça do Trabalho, estão listadas abaixo as facilidades e o seu ambiente de trabalho.

Principais facilidades criadas pelo SIP:

- Os dados são armazenados em um único banco de dados, o que permite sua recuperação e cruzamento para elaboração de estatísticas e prestação de informações ao público de forma extremamente

facilitada, respeitando sempre os critérios de segurança de acesso às informações;

- Cadastro separado para acesso distinto de partes e de advogados;
- Eliminação de livros de carga de processos, através do encaminhamento eletrônico e/ou guias emitidas pelo sistema;
- Pesquisa de processos através da combinação de informações;
- Emissão de certidões;
- Pesquisa de nomes por semelhança fonética;
- Controle de prazos e de execução de tarefas.

Ambiente:

- **Arquitetura Cliente/Servidor:** o SIP é um sistema que possui uma interface que trabalha em conjunto com os principais gerenciadores de banco de dados existentes no mercado.
- **Ambiente Multi-usuário:** a base de dados do sistema pode ser consultada e atualizada por múltiplos usuários simultaneamente, sendo a segurança e a integridade das informações armazenadas garantidas pelo SIP/PI.
- **Integração das Informações:** o SIP é um sistema desenvolvido em ambiente de Banco de Dados, dentro de uma filosofia de processamento integrado, onde uma informação é digitada uma única vez e fica disponível para qualquer módulo que precise fazer uso da mesma.
- **Segurança:** agregado ao SIP, existe um Módulo de Segurança, pelo qual o Administrador do Sistema autoriza ou não o acesso de usuários a funções ou informações específicas. Este módulo também é responsável pela auditoria no Sistema.
- **Impressão:** todos os relatórios podem ser visualizados na tela ou impressos em qualquer impressora disponível no Sistema Operacional;
- **Linguagem empregada:** a implementação do SIP foi desenvolvida utilizando a Linguagem de Programação Java. Esta é uma

característica importante, no que se refere ao ambiente, em virtude das vantagens do Java, já citadas.

4.1.1 Composição do Sistema

Conforme ilustrado na figura 20 abaixo, o sistema é composto por dez módulos que integram as funcionalidades do Sistema de Informação Processual.

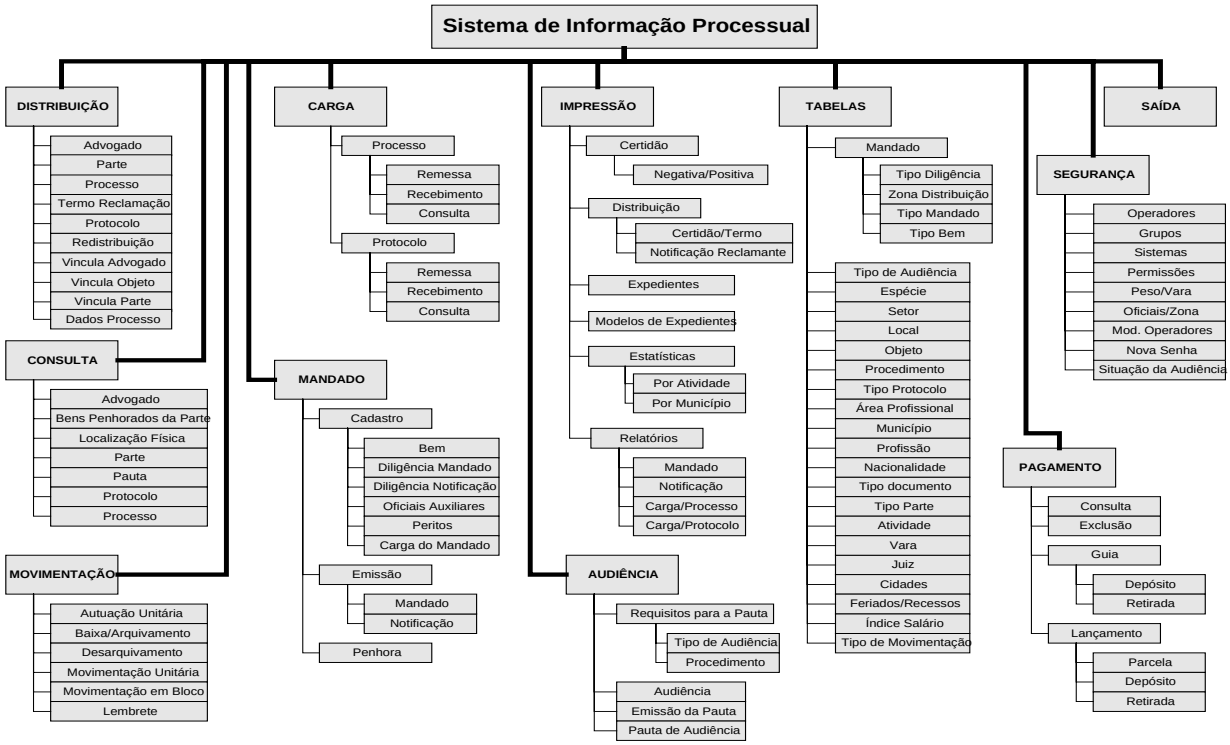

FIGURA 20 – Módulo do SIP.

Uma breve descrição e funcionalidade de cada módulo será relatado a seguir:

- **Módulo de Distribuição:** tem por objetivo realizar o cadastro dos processos judiciais, bem como das diversas partes envolvidas nesse processo, além de registrar a vinculação das partes do processo e seus respectivos advogados e a distribuição e redistribuição de processos por sorteios;

- **Módulo de Consulta:** propicia a realização de consultas a processos cadastrados, seus advogados e partes, assim como as pautas de audiência, bens penhorados de partes e a localização física de processos e protocolos;
- **Módulo de Movimentação Processual:** tem por objetivo realizar a movimentação unitária de processos, a movimentação de vários processos de forma simultânea, a baixa e a reativação de processos, a localização física de processos, assim como o controle da pauta de audiências;
- **Módulo de Carga:** visa realizar o controle de remessa, recebimento e consulta dos processos e mandados, possibilitando a eliminação dos livros de carga. Este controle se torna mais efetivo tanto no que diz respeito aos prazos de devolução, quanto às localizações;
- **Módulo de Mandados:** tem por objetivo permitir a realização da emissão e controle de qualquer mandado (ou notificação) utilizado no dia-a-dia do Cartório ou Vara e também permitir a utilização da Central de Mandados, Controle e Distribuição de Mandados/Notificação por Oficial de Justiça, além do Controle de Diligências, Emissão de Relatórios Diversos, Vinculação e Controle de Penhoras;
- **Módulo de Audiência:** visa permitir o suporte para os procedimentos de marcação de Pauta de Audiências e geração do “Termo de Audiência”, ou “Ata de Audiência”;
- **Módulo de Pagamento:** tem por objetivo permitir a definição dos tipos de recolhimentos por tipo de custas, como também a definição das regras de cálculo para cada tipo de recolhimento. Além disso, efetua o cálculo das custas para um processo, conforme os recolhimentos e regras pré-estabelecidos, realiza a atualização monetária dos valores históricos, efetua o cálculo de honorários de advogados e emite a conta de custas e guias de recolhimento;
- **Módulo de Impressão:** visa realizar a impressão de certidões, dados processuais, estatísticas por cartório, por comarca, por magistrado, por classe de processos e por tipo de movimentação, além de permitir que novas formas sejam definidas;

- **Módulo de Tabelas:** tem por objetivo realizar o cadastramento, alteração, consulta e exclusão de todas as tabelas básicas, visando o perfeito funcionamento do sistema;
- **Módulo de Segurança:** tem por objetivo realizar o cadastramento dos usuários do sistema e a respectiva lotação, assim como a liberação da autorização para acesso e atualização do banco de dados automaticamente. Este módulo será visto com mais detalhes a seguir.

4.2 Aplicação do Pacote de Assinatura Digital no Sistema de Informações Processuais

O pacote para a realização da assinatura digital foi desenvolvido pensando em diversas maneiras de utilização, facilitando assim sua implementação por parte do programador no Sistema de Informações Processuais (SIP), ou em outro sistema qualquer que se deseje implementar uma assinatura digital.

A tecnologia da assinatura digital e as facilidades dos serviços associados a ela serão realizadas acrescentando ao Módulo de Segurança do SIP, permitindo a autenticação informatizada dos documentos (processos, mandatos, certidões, entre outras). Ainda contará com os mecanismos de autorização e permissão que estão diretamente relacionados às propriedades de controle de acesso e disponibilidade de serviços, já implementadas no sistema atual, o que irá garantir a autenticidade e a integridade dos mesmos.

A figura 21, dá uma visão geral do funcionamento da assinatura digital integrado ao Módulo de Segurança do SIP, observando que o Módulo de Segurança define as permissões do usuário previamente cadastrado, restringindo o acesso aos recursos disponíveis no SIP, permitindo que cada usuário tenha acesso a conteúdos e ações específicas. Desta forma, nem todos os usuários do SIP terão acesso aos recursos totais ou parciais da assinatura digital, possibilitando, desta maneira, que um determinado usuário tenha permissão somente para verificar um arquivo assinado, enquanto outro a tenha para gerar o par de chaves necessárias para a

assinatura e ainda assinar um documento digitalmente, conforme as regras de segurança estabelecidas pelo administrador do SIP.

O pacote desenvolvido para a realização da assinatura digital possibilita que a assinatura do documento seja armazenada em um arquivo cujo nome seja igual ao arquivo que está sendo assinado, acrescido da extensão ".assinatura", ou também possibilita o armazenado em um banco de dados e, no momento da consulta (por usuários autorizados), possa ser realizada a verificação dessa assinatura.

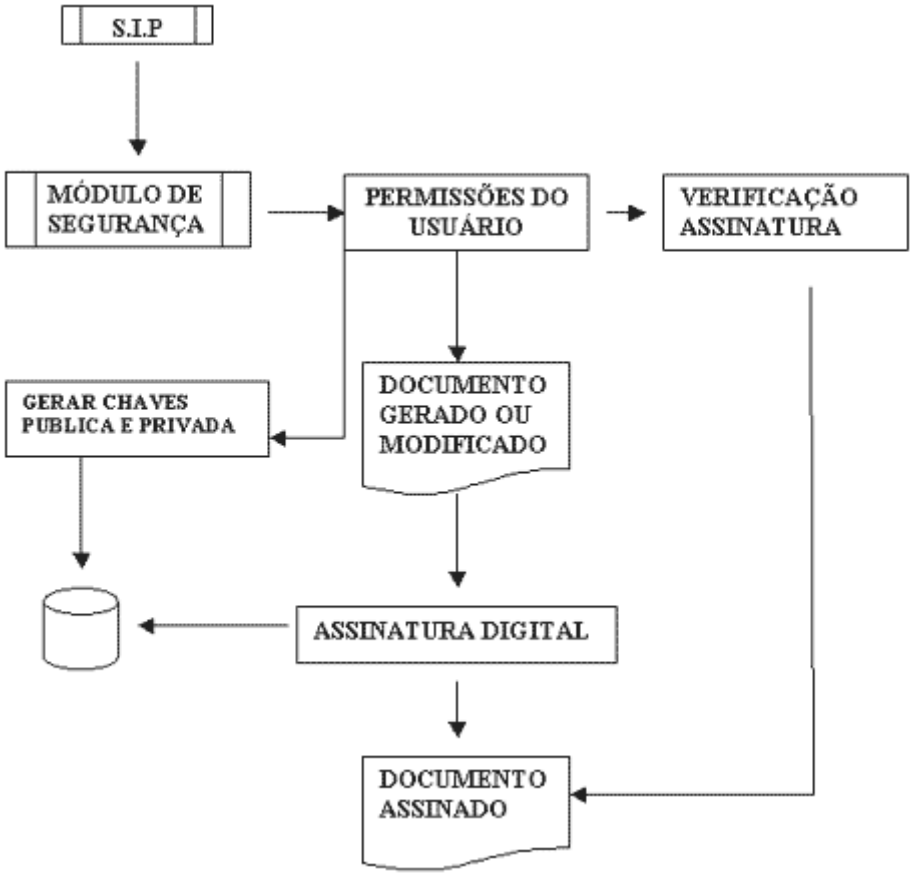

FIGURA 21 – Funcionamento da assinatura digital integrada ao SIP.

Para se acessar o SIP e ter acesso aos recursos do sistema disponível, com suas determinadas restrições de segurança, o usuário previamente cadastrado no Módulo de Segurança deverá informar o seu nome e sua respectiva senha. Atualmente, este processo é realizado em uma tela de identificação ilustrada na figura 22.

FIGURA 22 – Visualização da tela de identificação do sistema.

Caso os dados solicitados na tela de identificação estejam corretos, surge a tela principal do sistema (conforme figura 23, abaixo), porém o usuário que acabou de acessar o sistema somente terá acesso aos dados pré-definidos pelo administrador do SIP.

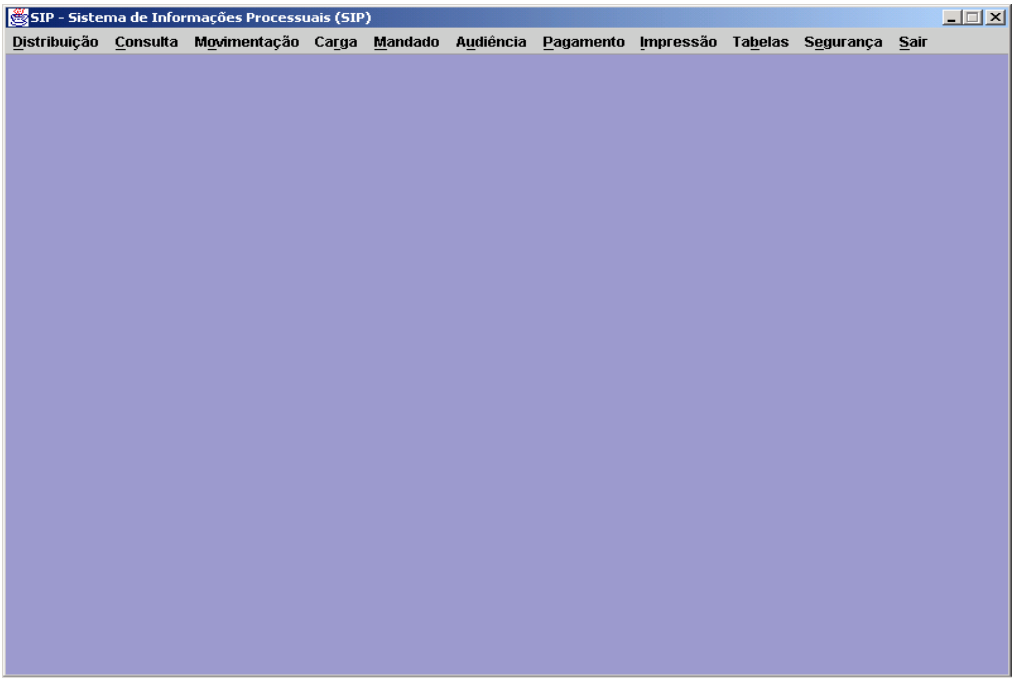

FIGURA 23 – Tela principal do SIP.

Para assinar digitalmente um documento, os usuários autorizados deverão, obrigatoriamente, gerar um par de chaves assimétricas, sendo uma chave privada (que deverá estar seguramente protegida) e outra chave pública (disponível). O processo de geração de chaves pelo usuário será tratado da mesma forma que as senhas de acesso ao sistema.

Para gerar este par de chaves assimétricas, o usuário deverá informar onde as chaves serão armazenadas e qual a frase senha para evitar a falsificação da assinatura digital. Esta frase senha deverá ser de uso pessoal de cada usuário e o mesmo deverá informar esta frase no momento em que estiver assinando digitalmente o arquivo para garantir a autenticidade.

As chaves geradas ficarão armazenadas em arquivos separados, contendo o mesmo nome, porém com as extensões diferenciadas, sendo que a chave pública terá a extensão “.pub” e a chave privada extensão “.pri”, possibilitando, desta forma, que somente o usuário que gerou o par de chaves possua a chave privada, e a chave pública poderá ser distribuída livremente para quem desejar verificar a autenticidade do arquivo assinado.

Quando o arquivo for assinado digitalmente, será necessário que a assinatura contenha alguns dados essenciais para possíveis consultas futuras. Pensando nisso, o pacote de assinatura digital desenvolvido contém uma classe chamada “dadosAssinatura”, que armazena e restaura dados como: o nome de quem está assinando o arquivo, data e hora da assinatura, nome e endereço “Ip” da máquina que realizou a assinatura e o local exato em que o arquivo assinado se encontrava no momento da assinatura. O armazenamento destes dados poderá ser realizado tanto em um banco de dados quanto no arquivo contendo a assinatura digital.

4.3 Forma de utilização do Pacote “Assinatura.jar”

Este componente computacional em forma de pacote de classes, foi desenvolvido pensando em diversas possibilidades de utilização. Uma das possibilidades destacadas é o armazenamento da assinatura digital em um arquivo, ou em um banco de dados. Como a criatividade dos desenvolvedores de sistemas computacionais pode ser quase ilimitada, neste capitulo será abordada cada classe contida no pacote de assinatura digital, encontrando-se no Anexo B, um programa fonte em Java, contendo algumas possibilidades de utilização deste pacote.

Para realizar a assinatura digital foi desenvolvido um pacote chamado “Assinatura.jar”, este pacote contém cinco classes, sendo elas:

- **Classe geraChaves:** utilizada para gerar um par de chaves assimétrica;
- **Classe assinatura:** responsável por assinar um arquivo digitalmente;
- **Classe verificaChaves:** verifica a frase senha digitada no processo de geração do par de chaves assimétrica;
- **Classe verificaAssinatura:** verifica se um arquivo foi assinado digitalmente ou se não houve alteração no arquivo após a assinatura digital;
- **Classe dadosAssinatura:** armazena e restaura os dados da assinatura digital.

Cada uma destas classes contém diversos métodos. Alguns deles são de uso da própria classe e são denominados “métodos privados”. Os demais foram desenvolvidos para a utilização do programador que irá desenvolver uma rotina de assinatura digital. Estes métodos são denominados “métodos públicos”. Como o intuito deste capitulo é demonstrar a forma de utilização deste pacote, somente serão demonstrados os “métodos públicos”, como veremos a seguir.

4.3.1 Classe geraChaves

É a classe em que será gerado o par de chaves necessárias para efetuar uma assinatura digital. Este par de chaves se define com uma chave pública e outra chave privada. Ao se gerar um par de chaves, estas serão armazenadas em forma de arquivos, sendo que o nome do arquivo do par de chaves gerado será igual para as duas chaves. O que diferencia visualmente a chave pública e a chave privada é a extensão do arquivo correspondente. Desta forma, o arquivo contendo a chave pública terá a extensão “.pub” e o arquivo contendo a chave privada terá a extensão “.pri”.

Cada uma destas chaves terá papéis diferenciados no decorrer do processo da assinatura digital, sendo que a chave privada será utilizada para efetuar a

assinatura digital propriamente dita. Esta chave será de domínio único do autor que está assinando o arquivo digitalmente, já a chave pública será utilizada para verificar a assinatura digital realizada. Ao contrário da chave privada, esta chave é de utilização pública, ou seja, pode ser distribuída para qualquer pessoa que queira verificar a validade do arquivo assinado digitalmente. Desta forma, se o arquivo assinado não foi alterado após a assinatura digital, a chave pública confirmará a autenticidade do documento. Esta classe contém o seguinte método público:

- **Método setGeraChaves (arquivo, seed)**

Este método é responsável por gerar o par de chaves para a assinatura digital. Para isto, basta passar como parâmetro o nome do arquivo que será gravado o par de chaves, e uma frase senha necessária para a segurança do arquivo que será assinado posteriormente. Quanto mais complexa for a frase senha, mais difícil será a falsificação da assinatura digital.

O parâmetro correspondente ao nome do arquivo deverá conter o diretório (local) onde o arquivo que se deseja assinar está localizado, mais o nome do arquivo e sua extensão. Este parâmetro pode ser passado somente de duas formas, ou no formato de string, utilizando java.lang.String ou no formato de arquivo, utilizando java.io.File. Já o parâmetro da frase senha só poderá ser passado no formato de string utilizando java.lang.String. Os exemplos a seguir ilustrarão a diferença de cada um deles.

Exemplo utilizando java.io.File:

```
assinaturaDigital.geraChaves gerar = new assinaturaDigital.geraChaves();
java.io.File arquivo  = new java.io.File("A:\\chave");
java.lang.String seed = "Frase Secreta";
boolean confere = gerar.setGeraChaves(arquivo, seed);
```

Exemplo utilizando java.lang.String:

```
assinaturaDigital.geraChaves gerar = new assinaturaDigital.geraChaves();
java.lang.String arquivo = "A:\\chave";
java.lang.String seed    = "Frase Secreta";
boolean confere = gerar.setGeraChaves(arquivo, seed);
```

Observem que a classe setGeraChaves retorna um valor booleano que é utilizado para verificar se foi gerado corretamente o par de chaves, ou seja, se o valor retornado for verdadeiro (*true*) significa que o par de chaves foi gerado e armazenado com sucesso.

4.3.2 Classe assinatura

Esta é a classe onde será realizada a assinatura digital propriamente dita. Existem duas possibilidades de uso desta classe para se realizar uma assinatura digital, sendo que a primeira consiste em usar o método getAssinatura. Este método retorna à assinatura digital em forma de um *array* de *bytes*, possibilitando o armazenamento alternativo da assinatura digital, como em um banco de dados, por exemplo. A segunda possibilidade de efetuar a assinatura digital é utilizando o método setAssinatura, onde a assinatura digital será armazenada em forma de arquivo. Este arquivo terá o nome do documento assinado mais a extensão ".assinatura", ou seja, um arquivo que está sendo assinado, cujo nome é "documento.doc" a sua assinatura digital será armazenada no arquivo com o nome "documento.doc.assinatura". Veremos com mais detalhes o funcionamento de cada um dos métodos públicos desta classe.

- **Método getAssinatura(arquivo, arqPrivada)**

Este método é referente à primeira possibilidade de se realizar a assinatura digital. Seus parâmetros são: local (diretório), mais o nome com a extensão do arquivo que se deseja assinar, e o local (diretório), mais o nome com a extensão do arquivo contendo a chave privada, gerada pela classe geraChaves. Estes parâmetros podem ser no formato de String utilizando java.lang.String ou no formato de arquivo, utilizando java.io.File, porém não é possível utilizar o método passando um parâmetro no formato de string e o outro parâmetro no formato de arquivo. Os dois parâmetros terão que conter o mesmo formato, como veremos nos exemplos a seguir.

Exemplo utilizando java.lang.String:

```
assinaturaDigital.assinatura assinar = new assinaturaDigital.assinatura();
java.lang.String arquivo    = "A:\\documento.doc";
java.lang.String arqPrivada = "A:\\chave.pri";
byte[] assinaturaByte  = assinar.getAssinatura(arquivo, arqPrivada);
```

Exemplo utilizando java.io.File:

```
assinaturaDigital.assinatura assinar = new assinaturaDigital.assinatura();
java.io.File arquivo    = new java.io.File("A:\\documento.doc");
java.io.File arqPrivada = new java.io.File("A:\\chave.pri");
byte[] assinaturaByte   = assinar.getAssinatura(arquivo, arqPrivada);
```

O método getAssinatura retorna um *array* de *bytes* contendo a assinatura digital do arquivo, possibilitando o armazenamento da assinatura digital de forma alternativa, como em um banco de dados.

- **Método setAssinatura(arquivo, arqPrivada)**

Esta é a segunda possibilidade de se realizar a assinatura digital. Este método contém os mesmos parâmetros do método getAssinatura, porém a grande diferença é que, ao invés de retornar um *array* de *bytes* contendo a assinatura digital, este retorna um valor booleano utilizado para validar a assinatura digital, ou seja, retorna verdadeiro (*true*) se a assinatura for efetuada e armazenada com sucesso. Este método também grava no mesmo local (diretório) do arquivo assinado a assinatura digital em forma de arquivo, onde o nome deste arquivo será igual ao do arquivo assinado, acrescentando a extensão “.assinatura”.

Assim como no método getAssinatura, os parâmetros também podem ser no formato de String, utilizando java.lang.String ou no formato de arquivo, utilizando java.io.File, porém não é possível utilizar o método passando um parâmetro no formato de string e outro no formato de arquivo, observem a seguir os exemplos deste método.

Exemplo utilizando java.io.File

```
assinaturaDigital.assinatura assinar = new assinaturaDigital.assinatura();
java.io.File arquivo    = new java.io.File("A:\\documento.doc");
java.io.File arqPrivada = new java.io.File("A:\\chave.pri");
boolean confere = assinar.setAssinatura(arquivo, arqPrivada);
```

Exemplo utilizando java.lang.String

```
assinaturaDigital.assinatura assinar = new assinaturaDigital.assinatura();
java.lang.String arquivo    = "A:\\documento.doc";
java.lang.String arqPrivada = "A:\\chave.pri";
boolean confere = assinar.setAssinatura(arquivo, arqPrivada);
```

4.3.3 Classe verificaChaves

Esta classe tem como finalidade fazer uma comparação entre a frase senha digitada no processo de geração do par de chaves, realizada pela classe *geraChaves,* com uma frase passada como parâmetro. Um dos motivos de se utilizar este método é o fato de poder comparar se quem gerou um par de chaves (mais especificamente a chave privada) e o autor da assinatura digital de um determinado arquivo são a mesma pessoa, pois se este autor não souber qual foi a frase senha utilizada no processo de geração do par de chaves, terá como impedi-lo de realizar a assinatura digital. A seguir veremos como utilizar o método setVerificaChaves desta classe.

- **Método setVerificaChaves(arquivo, seed)**

Este método realiza a comparação da frase senha utilizada no processo de geração do par de chaves, realizada pela classe geraChaves, passando como parâmetro o local (diretório), mais o nome e extensão do arquivo contendo a chave privada e a frase senha que se quer comparar. O parâmetro correspondente ao nome do arquivo pode ser passado em forma de string, utilizando java.lang.String ou em forma de arquivo, utilizando java.io.File. Já o parâmetro correspondente à frase senha que se quer comparar somente poderá ser passada em forma de string, utilizando java.lang.String, como ilustram os exemplos a seguir.

Exemplo utilizando java.io.File:

```
assinaturaDigital.verificaChaves chave = new assinaturaDigital.verificaChaves();
java.io.File arquivo  = new java.io.File("A:\\chave.pri");
java.lang.String seed = "Frase Secreta";
boolean confere = chave.setVerificaChaves(arquivo, seed);
```

Exemplo utilizando java.lang.String:

```
assinaturaDigital.verificaChaves chave = new assinaturaDigital.verificaChaves();
java.lang.String arquivo  = "A:\\chave.pri";
java.lang.String seed = "Frase Secreta";
boolean confere = chave.setVerificaChaves(arquivo, seed);
```

Quando se utiliza este método, ele retorna um valor booleano que indica a validade da frase senha. Sendo assim, se a frase senha utilizada na geração do par de chaves for a mesma frase passada como parâmetro, será retornado um valor verdadeiro (*true*).

4.3.4 Classe verificaAssinatura

Quando se deseja verificar se um determinado documento foi assinado digitalmente, ou ainda se este documento não foi mais modificado após a assinatura digital, esta classe irá realizar o procedimento. A utilização desta classe é bem simples, porém deve-se ficar atento a um simples detalhe. O método setVerificaAssinatura aqui possibilita a verificação da assinatura digital de duas maneiras. A primeira delas acontece passando dois parâmetros, onde o primeiro é o local (diretório), mais o nome com extensão do arquivo que está verificando a autenticidade da assinatura digital. O segundo parâmetro é o local (diretório), mais o nome com extensão do arquivo contendo a chave pública gerada pela classe geraChaves. A segunda maneira de utilização do método setVerificaAssinatura é passando três parâmetros cujo primeiro deles é um array de bytes correspondente a uma assinatura digital que foi armazenada de forma alternativa, como em um banco de dados por exemplo, o segundo parâmetro é o local (diretório), mais o nome com extensão do arquivo que esta verificando a autenticidade da assinatura digital, e o terceiro parâmetro é o local (diretório), mais o nome com extensão do arquivo

contendo a chave pública, gerada pela classe geraChaves. A seguir veremos com mais detalhes a utilização do método setVerificaAssinatura da classe verificaAssinatura.

- **Método setVerificaAssinatura(arquivo, arqPublica)**

Verifica a assinatura digital de um arquivo assinado, passando somente dois parâmetros, onde o primeiro parâmetro é o local (diretório), mais o nome com extensão do arquivo que está verificando a autenticidade da assinatura digital, e o segundo parâmetro é o local (diretório), mais o nome com extensão do arquivo contendo a chave pública gerada pela classe geraChaves. Estes dois parâmetros podem ser no formato de String, utilizando java.lang.String ou no formato de arquivo, utilizando java.io.File. Porém, não é possível utilizar este método passando um parâmetro no formato de string, e o outro parâmetro no formato de arquivo, pois ambos têm que ter o mesmo formato, como veremos nos exemplos a seguir.

Exemplo utilizando java.io.File:

```
assinaturaDigital.verificaAssinatura verificar = new assinaturaDigital.verificaAssinatura();
java.io.File arquivo    = new java.io.File("A:\\documento.doc");
java.io.File arqPublica = new java.io.File("A:\\chave.pub");
boolean ok = verificar.setVerificaAssinatura(arquivo, arqPublica);
```

Exemplo utilizando java.lang.String:

```
assinaturaDigital.verificaAssinatura verificar = new assinaturaDigital.verificaAssinatura();
java.lang.String arquivo    = "A:\\documento.doc";
java.lang.String arqPublica = "A:\\chave.pub";
boolean ok = verificar.setVerificaAssinatura(arquivo, arqPublica);
```

- **Método setVerificaAssinatura(assinatura, arquivo, arqPublica)**

Verifica a Assinatura Digital do arquivo assinado, passando três parâmetros, onde o primeiro parâmetro é um array de byte contendo a assinatura digital (normalmente estes bytes estavam gravados de forma alternativa, como em um banco de dados, por exemplo), o segundo é o local (diretório), mais o nome com extensão do arquivo que está verificando a autenticidade da assinatura digital, e o terceiro parâmetro é o local (diretório), mais o nome com extensão do arquivo

contendo a chave pública gerada pela classe geraChaves. A utilização do método com os três parâmetros se deve ao fato de que a classe assinatura possibilita o armazenamento da assinatura digital de forma alternativa. Portanto, quando se utiliza este recurso, só será possível verificar a assinatura digital passando como parâmetro o array de bytes armazenado de forma também alternativa. O formato do primeiro parâmetro somente poderá ser um array de bytes contendo a assinatura digital, já os outros dois parâmetros podem ser um arquivo, utilizando java.ioFile, ou uma string, utilizando java.lang.String. Porém, assim como na utilização deste método com passagem de dois parâmetros, não é possível utilizá-lo passando um parâmetro no formato de string, e o outro parâmetro no formato de arquivo, já que o segundo e o terceiro parâmetros têm que ter o mesmo formato, como veremos nos exemplos a seguir.

Exemplo utilizando java.io.File:

```
assinaturaDigital.verificaAssinatura verificar = new assinaturaDigital.verificaAssinatura();
byte[] assByte = new byte[8192];
assByte = (Array de Byte);
java.io.File arquivo = new java.io.File("A:\\documento.doc");
java.io.File arqPublica = new java.io.File("A:\\chave.pub");
boolean ok = verificar.setVerificaAssinatura(assByte, arquivo, arqPublica);
```

Exemplo utilizando java.lang.String:

```
assinaturaDigital.verificaAssinatura verificar = new assinaturaDigital.verificaAssinatura();
byte[] assByte = new byte[8192];
assByte = (Array de Byte);
java.lang.String arquivo    = "A:\\documento.doc";
java.lang.String arqPublica = "A:\\chave.pub";
boolean ok = verificar.setVerificaAssinatura(assByte, arquivo, arqPublica);
```

4.3.5 Classe dadosAssinatura

Esta classe foi acrescentada no pacote “Assinatura.jar”, visando a necessidade de se obter alguns dados necessários referentes à assinatura digital, como por exemplo, o nome do autor, data e hora da realização da assinatura digital, dentre outros dados que veremos com mais detalhes na descrição dos métodos desta classe. Porém, para se obter estes dados, os mesmos devem ser

primeiramente definidos e armazenados no momento que se realiza o processo da assinatura digital.

O objetivo desta classe é justamente definir e/ou obter estes dados de maneira simples e direta, sendo que, quando se deseja defini-los, deve se levar em consideração que este processo deverá ser efetuado antes de realizar a assinatura digital, pois os dados aqui estipulados deverão ser armazenados no momento em que se assina o arquivo digitalmente. Já para obter os dados de um documento assinado, basta informar o local (diretório), nome e extensão do arquivo deste documento.

A nomenclatura dos métodos desta classe é de fácil utilização, pois segue os padrões da maioria dos métodos utilizados na linguagem Java. Desta forma, quando se desejar utilizar algum método referente à definição dos dados da assinatura, é aconselhável que este método inicia-se com a palavra “set”, seguido pelo nome do dado que se deseja definir, utilizando a primeira letra do nome deste dado em maiúsculo. E, quando se deseja obter algum dado, é aconselhável que o nome do método se inicia com a palavra “get”, seguido pelo nome do dado que se deseja obter, utilizando a primeira letra do nome deste dado em maiúsculo, por exemplo: utiliza-se o método setAutor para definir, e o método getAutor para obter o nome do autor da assinatura digital. A seguir veremos a utilização dos métodos públicos desta classe com mais detalhes.

Veremos primeiramente como se utiliza a classe dadosAssinatura para definir os dados necessários à assinatura digital. O primeiro passo a ser dado neste processo é estipular o diretório (local) mais o nome do arquivo que está sendo assinado com sua extensão. Este processo poderá ser feito de duas maneiras: a primeira delas é passando o diretório (local), nome e extensão do arquivo como parâmetro no momento em que está instanciando a classe dadosAssinatura. A segunda opção utiliza o método setPath. Os exemplos contidos nos métodos de definição dos dados da assinatura digital utilizarão apenas a primeira, pois a segunda opção será melhor explicada na definição do método setPath.

- **Método setAutor(java.lang.String nAutor)**

Define um nome para o autor que está assinando o arquivo digitalmente, bastando, para isso, apenas passar como parâmetro uma String, utilizando

java.lang.String, contendo o nome do autor da assinatura digital, como ilustra o exemplo a seguir.

```
java.lang.String arquivo = "A:\\documento.doc";
java.lang.String nAutor  = "Nome do Autor da Assinatura";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
dados.setAutor(nAutor);
```

- **Método setData(java.lang.String nData)**

Define uma data para a assinatura digital. Este método possui duas maneiras de ser utilizado. A primeira delas é utilizar este método sem passar nenhum parâmetro. Desta forma, o método define a data do computador que está realizando a assinatura digital como sendo a data em que a assinatura está sendo realizada. A segunda opção diz respeito ao fato de se desejar que seja definida uma determinada data para a assinatura digital, bastando para isso, passar um parâmetro em forma de String, utilizando java.lang.String, contendo a data desejada. Esta opção é normalmente utilizada quando obtém a data de outra forma que não seja a do computador que está efetuando a assinatura digital, como a data de um servidor ou de um banco de dados. Observem nos exemplos a seguir a utilização das duas formas deste método:

Definindo a data do computador que esta realizando a assinatura digital.

```
java.lang.String arquivo = "A:\\documento.doc";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
dados.setData();
```

Definindo uma data alternativa da assinatura digital, como passagem de parâmetro.

```
java.lang.String arquivo = "A:\\documento.doc";
java.lang.String nData   = "01/01/2004";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
dados.setData(nData);
```

- **Método setHora(java.lang.String nHora)**

Define uma hora para a assinatura digital. Assim como o método setData, este método também possui duas maneiras de ser utilizado, bem semelhantes ao método anterior. A primeira delas é a utilização deste método sem passar parâmetro algum, pois desta forma, é definida a hora do computador que está realizando a

assinatura digital. A segunda opção é utilizada caso deseje que seja definida uma determinada hora para a assinatura digital que não seja a do computador que está realizando a assinatura, bastando, para isso, passar um parâmetro em forma de String, utilizando java.lang.String e contendo a hora desejada. Esta opção é normalmente utilizada quando obtém a hora de outra forma que não seja a do computador que está efetuando a assinatura digital, como a hora de um servidor ou de um banco de dados, por exemplo. Observem nos exemplos a seguir a utilização das duas formas deste método.

Definindo a hora do computador que esta realizando a assinatura digital.

```
java.lang.String arquivo = "A:\\documento.doc";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
dados.setHora();
```

Definindo uma hora alternativa da assinatura digital, como passagem de parâmetro.

```
java.lang.String arquivo = "A:\\documento.doc";
java.lang.String nHora   = "20:45:22";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
dados.setHora(nHora);
```

- **Método setDefault(java.lang.String nAutor)**

Este método foi adicionado na classe dadosAssinatura como um facilitador, pois, com a passagem de apenas um parâmetro contendo o nome do autor, se define três dados referentes à assinatura digital. O parâmetro passado deverá ser em forma de String, utilizando java.lang.String e contendo o nome do autor da assinatura digital, sendo definido, além do autor, a data e a hora do computador em que está realizando a assinatura.

```
java.lang.String arquivo = "A:\\documento.doc";
java.lang.String nAutor  = "Nome do Autor da Assinatura";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
dados.setDefault(nAutor);
```

- **Método setIp()**

Seta o endereço de IP do computador que está assinando o arquivo digitalmente.

```
java.lang.String arquivo = "A:\\documento.doc";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
dados.setIp();
```

- **Método setMaquina()**

Seta o nome do computador que está assinando o arquivo digitalmente.

```
java.lang.String arquivo = "A:\\documento.doc";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
dados.setMaquina();
```

- **Método setPath(java.lang.String nPath)**

Especifica o local e o nome do arquivo que está sendo assinado digitalmente, passando como parâmetro uma String, utilizando java.lang.String e contendo o diretório (caminho), o nome e extensão do arquivo. Observe, no exemplo a seguir, que não é passado nenhum parâmetro quando se instância a classe dadosAssinatura. Mas não há nenhum problema em se realizar este procedimento, devendo-se apenas ficar atento ao fato de que, se utilizar as duas formas de se especificar o local e o nome do arquivo que está sendo assinado, o parâmetro que prevalecerá é o passado através do método setPath.

```
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura();
java.lang.String arquivo = "A:\\documento.doc";
dados.setPath(arquivo);
```

Agora veremos a utilização da classe dadosAssinatura para obter os dados da assinatura digital de um arquivo já assinado. Assim como para definir os dados da assinatura, quando se deseja obtê-los, também será necessário estipular primeiramente o diretório (local), mais o nome do arquivo que está sendo assinado

com sua extensão. Este processo poderá ser feito de duas maneiras, onde a primeira passará o diretório (local), mais o nome com extensão do arquivo como parâmetro, no momento em que está instanciando a classe dadosAssinatura. A segunda opção utilizará o método setPath. Os exemplos contidos nos métodos de aquisição dos dados da assinatura digital utilizarão a primeira opção, pois a segunda já foi mencionada na definição do método setPath.

- **Método getAutor()**

Obtém o nome do autor que efetuou a assinatura digital. Este nome é retornado pelo método em forma de String, utilizando java.lang.String. Caso não tenha estipulado nenhum autor quando o arquivo foi assinado, é retornado uma String vazia.

```
String arquivo = "A:\\documento.doc.assinatura";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
String autor = dados.getAutor();
System.out.println("Autor da Assinatura: "+autor);
```

- **Método getDados()**

Obtém uma frase em forma de String, utilizando java.lang.String, contendo todos os dados do arquivo assinado de forma codificada e embaralhada. Este método é utilizado pela classe assinatura do pacote Assinatura.jar de maneira direta, ou seja, não precisa da interferência de terceiros para executá-lo, pois é esta frase codificada que será armazenado no arquivo de assinatura digital no momento da assinatura do arquivo. Porém este método também poderá ser utilizado a qualquer momento que se deseje obter todos os dados gravados na assinatura digital, de maneira embaralhada e codificada, como no exemplo a seguir.

```
String arquivo = "A:\\documento.doc.assinatura";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
String dadosCompletos = dados.getDados();
```

- **Método getDadosBanco()**

Obtém uma frase em forma de String, utilizando java.Lang.String, contendo todos os dados do arquivo assinado de forma original, separados apenas pelo caractere “|” (*Pipeline*). A utilização deste método é necessária quando se deseja armazenar os dados da assinatura de forma alternativa, normalmente um Banco de Dados.

```
String arquivo = "A:\\documento.doc.assinatura";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
String dadosCompletos = dados.getDadosBanco();
```

- **Método getData()**

Obtém a data em que o arquivo foi assinado digitalmente. O método retorna uma String, utilizando java.lang.String, contendo a data em que o arquivo foi assinado. Caso não foi estipulada uma data no momento da assinatura, este método retornará a string em vazio.

```
String arquivo = "A:\\documento.doc.assinatura";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
String data = dados.getData();
System.out.println("Data da Assinatura: "+data);
```

- **Método getHora()**

Obtém a hora em que o arquivo foi assinado digitalmente. O método retorna uma String, utilizando java.lang.String, contendo a hora em que o arquivo foi assinado. Caso não tenha sido estipulada uma hora no momento da assinatura, este método retornará a string em vazio.

```
String arquivo = "A:\\documento.doc.assinatura";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
String hora = dados.getHora();
System.out.println("Hora da Assinatura: "+hora);
```

- **Método getIp()**

Obtém o endereço de IP do computador que assinou o arquivo digitalmente. O método retorna uma String, utilizando java.lang.String, contendo o endereço de Ip do computador em que o arquivo foi assinado. Caso não tenha sido especificado nenhum endereço de Ip, no momento em que o arquivo foi assinado, este método retornará a string em vazio.

```
String arquivo = "A:\\documento.doc.assinatura";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
String ip = dados.getIp();
System.out.println("Endereço IP do Computador que Realizou a Assinatura: "+ip);
```

- **Método getMaquina()**

Obtém o nome do computador que assinou o arquivo digitalmente. O método retorna uma String, utilizando java.lang.String, contendo o nome do computador em que o arquivo foi assinado. Caso não tenha sido definido nenhum nome para o computador no momento em que o arquivo foi assinado, este método retornará a string em vazio.

```
String arquivo = "A:\\documento.doc.assinatura";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
String maquina = dados.getMaquina();
System.out.println("Nome do Computador que Realizou a Assinatura: "+maquina);
```

- **Método getPath()**

Obtém o nome do arquivo assinado digitalmente. O método retorna uma String, utilizando java.lang.String, contendo o diretório (local), mais o nome com extensão do arquivo quando ele foi assinado.

```
String arquivo = "A:\\documento.doc.assinatura";
assinaturaDigital.dadosAssinatura dados = new assinaturaDigital.dadosAssinatura(arquivo);
String path = dados.getPath();
System.out.println("Diretório e Nome do Arquivo Assinado: "+path);
```

4.4 Interface sugerida utilizando o Pacote “Assinatura.jar”

A interface sugerida neste capítulo tem o intuito de ilustrar os processos relacionados à assinatura digital, podendo cada programador desenvolver a sua própria interface. Os códigos fonte em Java utilizado para desenvolver as telas de interface sugerida, encontram-se no Anexo B.

A figura 24 abaixo, é a tela principal do modelo de interface sugerido para a utilização do pacote “Assinatura.jar”

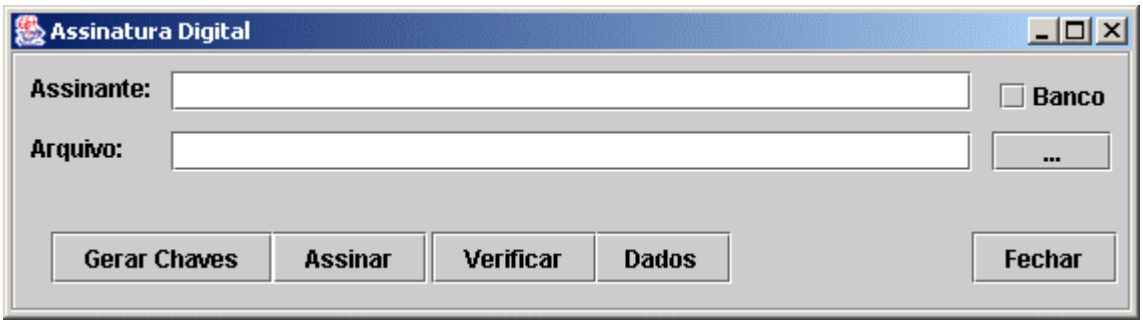

FIGURA 24 – Tela principal da Assinatura Digital.

O campo “Assinante:” é usado para especificar o nome de quem esta assinando o documento, e o campo “Arquivo:” é onde se especifica o nome do documento que deseja ser assinado ou verificado. Este campo deve conter o diretório onde o documento se encontra mais o nome do arquivo. O botão “...” localizado à direita deste campo serve para facilitar a procura do documento.

Cada botão localizado abaixo do campo “Arquivo:” ilustrado na figura 24 tem uma função especifica:

- **Gerar Chaves**: Este botão serve para iniciar o processo de geração da chave privada e da chave pública utilizada no processo de assinatura digital.
- **Assinar**: Serve para iniciar o processo de assinatura digital do documento especificado.
- **Verificar**: É utilizado para verificar a assinatura do documento especificado.

- **Dados**: Mostra os dados armazenados no processo de assinatura digital.
- **Fechar**: Fecha a tela ilustrada pela figura 24, encerrando o programa de Assinatura Digital desenvolvido utilizando o pacote “Assinatura.jar”

Ao clicar com o mouse sobre o botão “Gerar Chaves”, será mostrada uma tela conforme a ilustrada pela figura 25 abaixo, onde o usuário deve especificar uma frase secreta para dar inicio ao processo de geração da chaves privada e da chave pública.

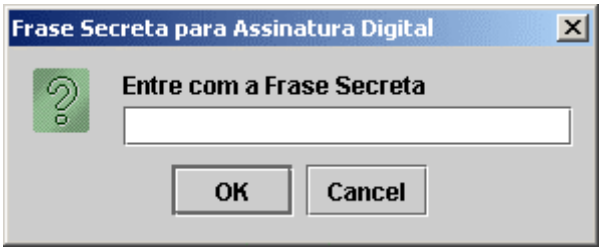

FIGURA 25 – Tela Frase Secreta do botão “Gerar Chaves”.

Após o usuário digitar uma frase secreta e clicar no botão “OK”, o sistema disponibilizará uma tela ilustrada pela figura 26, que serve para o usuário especificar o nome das chaves, e o local onde elas serão armazenadas. Lembrando que o nome da chave pública e da chave privada é o mesmo, diferenciado apenas pela extensão, sendo que a chave pública a extensão é “.pub” e a chave privada a extensão é “.pri”. Neste modelo, o sistema coloca automaticamente as extensões equivalentes, sendo necessário o usuário especificar somente um nome para as duas chaves no campo “File Name:”.

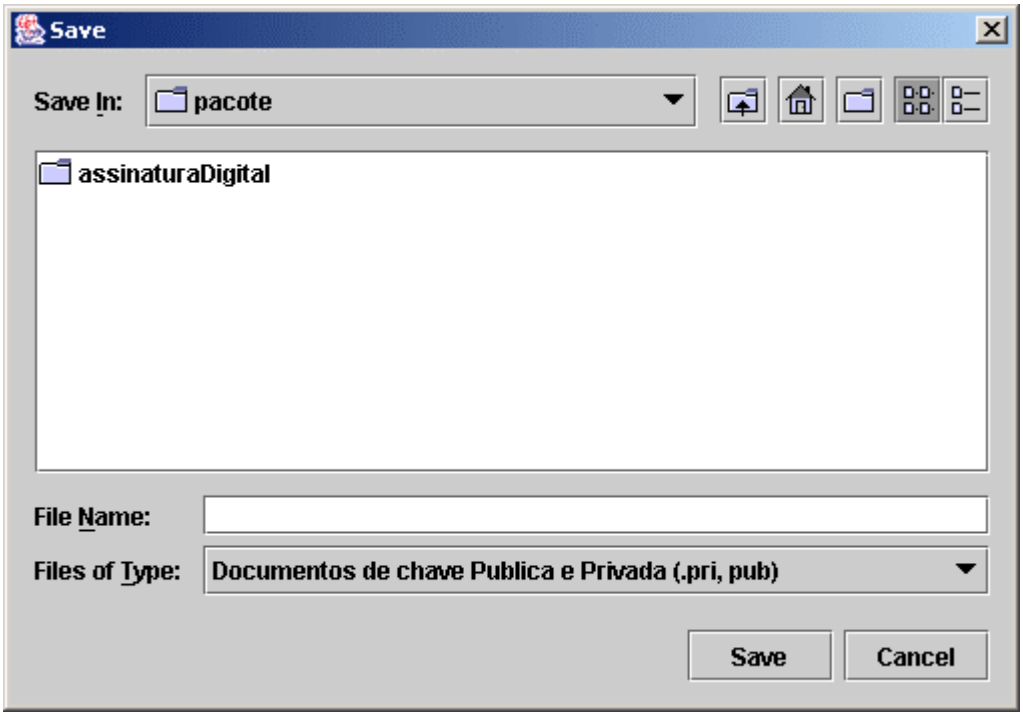

FIGURA 26 – Tela de gravação das chaves privada e pública.

Após o usuário especificar o nome das chaves e clicar no botão “Save”, o sistema disponibilizará uma mensagem confirmando a geração do par de chaves, o local onde as chaves foram armazenadas e o nome das chaves com sua respectiva extensão, conforme demonstra a figura 27 abaixo.

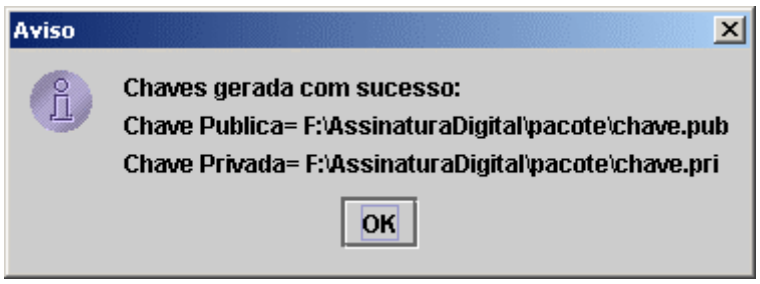

FIGURA 27 – Tela de confirmação da geração das chaves privada e pública.

Para assinar um documento digitalmente, após a geração do par de chaves, o usuário deve especificar no campo “Arquivo:” o local mais o nome do documento que será assinado, e clicar com o mouse sobre o botão “Assinar”, ambos na tela principal ilustrada pela figura 24. Seguindo estes procedimentos, o sistema disponibilizara uma tela ilustrada pela figura 28 a seguir, onde o usuário deve especificar a chave privada.

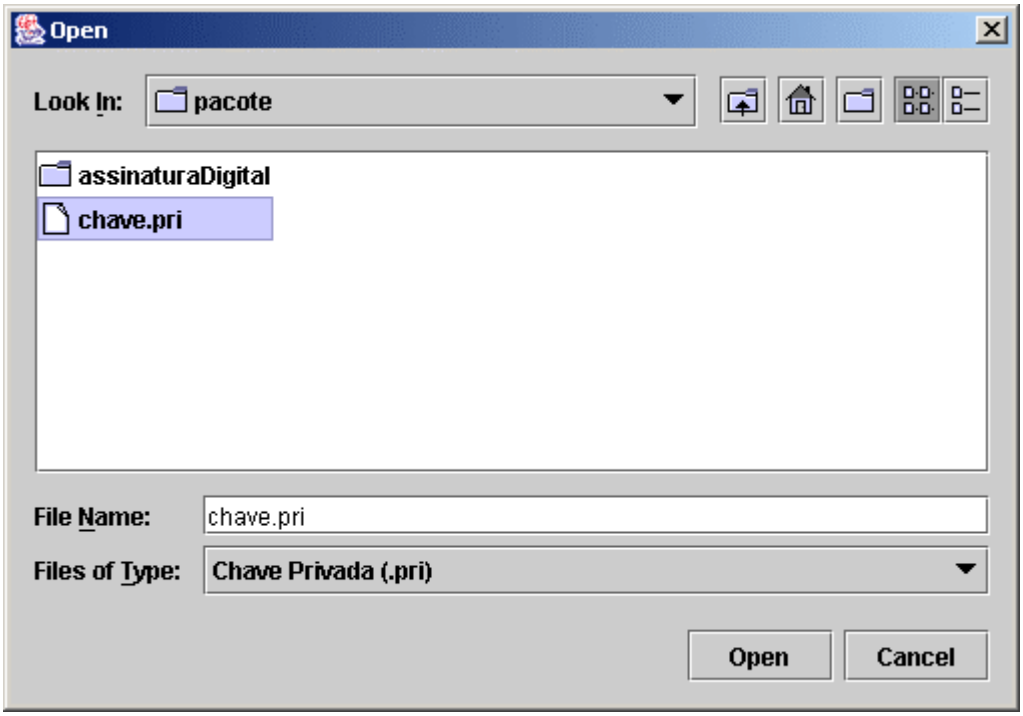

FIGURA 28 – Tela de especificação da chave privada.

Após o usuário especificar a chave privada para realizar a assinatura digital, o sistema disponibilizará uma tela ilustrada pela figura 29 abaixo, onde o usuário deve confirmar a frase secreta utilizada no processo de geração do par de chaves. Caso a frase secreta seja a mesma utilizada no processo de geração do par de chaves, e o nome do arquivo que será assinado for válido, o sistema confirmará que o documento foi assinado com sucesso.

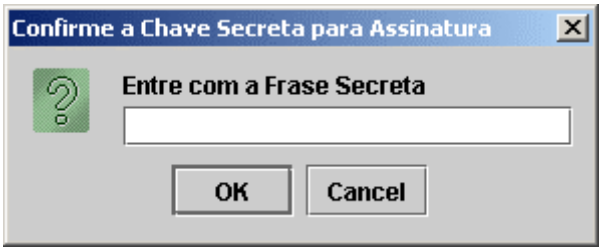

FIGURA 29 – Tela de confirmação da frase secreta.

Para verificar a assinatura digital de um documento, o usuário deve especificar no campo “Arquivo:” o local mais o nome do documento a ser verificado, e clicar com o mouse sobre o botão “Verificar”, ambos na tela principal ilustrada pela

figura 24. Seguindo estes procedimentos, o sistema disponibilizará uma tela ilustrada pela figura 30 abaixo, onde o usuário deve especificar a chave pública.

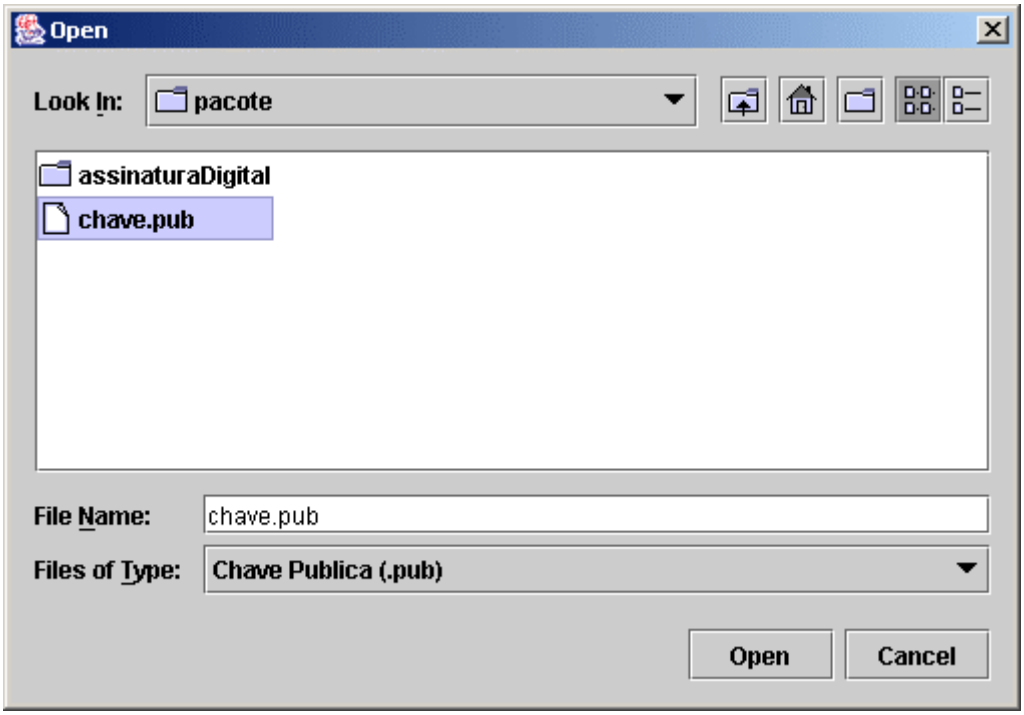

FIGURA 30 – Tela de especificação da chave pública.

Caso a chave pública especificada for válida e o documento assinado digitalmente não sofreu nenhuma alteração, o sistema confirmará que a assinatura digital do documento especificado é válida.

Para obter os dados armazenados referente à assinatura digital de um documento, o usuário deve especificar no campo “Arquivo:” o local mais o nome do documento que deseja obter os dados, e clicar com o mouse sobre o botão “Dados”, ambos na tela principal ilustrada pela figura 24. Após estes procedimentos, o sistema disponibilizará os dados da assinatura digital conforme ilustrado pela figura 31, onde os campos demonstrados correspondem aos dados armazenados no momento que o documento foi assinado, onde o campo “Autor=” é o nome do assinante do documento, “Data=” é a data da assinatura do documento, “Horas=” é a hora que o documento foi assinado, “Arquivo=” é o local mais o nome do arquivo assinado, “Maquina=” é o nome do computador que assinou o documento, e o “Ip=” é o endereço de Ip do computador que assinou o documento.

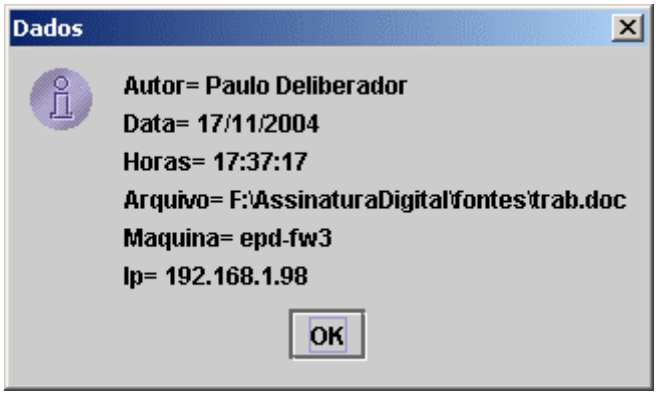

FIGURA 31 – Tela dos dados da Assinatura Digital.

Caso o usuário deseje assinar, verificar ou obter os dados da assinatura digital em um banco de dados, basta marcar o campo “Banco” no canto superior direito da tela ilustrada pela figura 24.

5 CONCLUSÃO

Nesta dissertação foi visto que para implementar uma assinatura digital, deve-se levar em consideração vários fatores, desde os algoritmos de criptografia, passando pela escolha de uma linguagem de programação para implementar e pela utilização destes algoritmos, até os processos legislativos para regularizar as assinaturas digitas, tornado-as tão válidas quanto às assinaturas comuns do papel e caneta.

Após os estudos realizados ao longo deste trabalho, apontando os aspectos legais e os principais algoritmos de criptografia, com a finalidade de formar um suporte teórico sobre o funcionamento da assinatura digital, foi possível propor um componente computacional em forma de pacote (*package*), capaz de auxiliar os analistas de sistemas e programadores a desenvolverem aplicativos utilizando as técnicas de assinatura digital, incorporando-as ao Sistema de Informatização Processual (SIP).

Através deste componente computacional, é possível implementar as técnicas de assinatura digital em diversos módulos do SIP, aperfeiçoando a segurança do sistema, possibilitando que seus usuários possam assinar digitalmente todo e qualquer trâmite processual e posteriormente realizar a verificação de validade da assinatura inserida. Sendo assim, o desenvolvedor do módulo desejado pode utilizar este componente tanto para aplicações locais (servidor / servidor), quanto para aplicações remotas (cliente / servidor) através de uma rede de comunicação. Porém, para isso se tornar possível, este componente foi desenvolvido seguindo aos seguintes critérios:

- A linguagem de programação Java, além das principais características e facilidades identificadas na implementação de um processo de assinatura digital, é a mesma linguagem utilizada pela equipe de desenvolvimento do SIP, facilitando a integração e utilização do componente computacional, pois a equipe de desenvolvimento já está familiarizada com esta linguagem.
- As nomenclaturas utilizadas nos métodos do componente, são de fácil entendimento para desenvolvedores que utilizam a linguagem Java,

pois procura seguir na maioria das vezes o mesmo padrão que a Sun Microsystems utiliza no desenvolvimento de seus métodos.

- Utiliza em seu contexto, algoritmos de assinatura, geração do par de chaves e de verificação da assinatura considerados seguros pelos criptógrafos, livres de patente e de fácil implementação em Java.

Este componente computacional foi desenvolvido para ser utilizado pelos desenvolvedores do SIP, mais também pode ser utilizado por qualquer desenvolvedor Java que deseja incorporar as técnicas de assinatura digital em suas aplicações. Com a utilização do componente desenvolvido, estas aplicações assim como o SIP, poderão contar com a segurança proposta pela assinatura digital, garantindo a autoria e a integridade dos documentos assinados, tornando as aplicações mais seguras e preparadas para a necessidade crescente deste tipo de segurança imposta atualmente pela *internet*, ou por qualquer outro meio de troca de informações digitais.

5.1 Recomedações para trabalhos futuros

Como trabalho futuro, recomenda-se um estudo crescente das técnicas de criptografia e assinatura digital, pois é cada vez maior o número de pessoas, instituições públicas e privadas que necessitam deste tipo de segurança. Nos dias atuais, os algoritmos apresentados nesta dissertação, são considerados seguros pelos criptógrafos, mas com o decorrer dos tempos, novas fraquezas poderão ser descobertas, ou computadores mais velozes poderão ser fabricados, tornando estes algoritmos mais vulnerável a ataques por "força bruta”, ou qualquer outro tipo de quebra de segurança. No entanto, estudos referentes a novos algoritmos de criptografia e mecanismos de segurança em sistemas computacionais serão de uma importância significativa, e na medida do possível, tornar estes estudos públicos, para que seus resultados sejam acessíveis para um número maior de pessoas.

REFERÊNCIAS

ABRANTES,A. Souza de. **Patentes de Programas de Computador: Um Estudo Dos Fundamentos de Exame E Análise De Estatísticas Do Setor,** 2002. Disponível em:<http://www.nepi.adv.br/doutrina/patentes_programas.htm>. Acesso em: 12 de out. 2003.

ADAMS, Carlisle; LLOYD, Steve **Understanding Public-key Infrastructure**: Concepts, Standards, and Deployment Considerations. New Riders Publishing, Indianapolis, IN, USA, Novembro de 1999.

ALBERTIN, Alberto Luiz. **Comércio Eletrônico: Modelo, Aspectos e Contribuições de sua Aplicação**. São Paulo: Atlas, 1999.

ARNOLD, Ken, GOSLING, James. **Programando em Java.** São Paulo: Makron Books, 1997.

BARBOSA, Manoel Bernardo. **Criptografia Aplicada.** 2003. Disponível em <http://www.di.uminho.pt/~glmf/ca/> Acesso em 17 de Jan. 2004.

BERNSTEIN, Terry, BRIMANI, Ansh B. SHULTZ, Eugene, SIEGEL, Carol A. **Segurança na Internet.** Rio de Janeiro: Campus, 1997.

BITTENCOURT Angela. **E-Commerce.** 2001. Disponível em: <http://www.e-commerce.org.br/Artigos_Geral/assinatura_digital.htm> Acesso em: 18 de Ago. 2004.

BITTENCOURT, Ângela. **Assinatura Digital não é Assinatura Formal.** 2002. Disponível em: <//www.modulo.com.br >. Acesso em: 02 de set. 2004.

BURNETT, Steve; PAINE, Stephe. **Criptografia e Segurança: O Guia Oficial RSA**. Rio de Janeiro: Campus, 2002.

CBEJI, **Centro Brasileiro de Estudos Jurídicos da Internet.** 2002. Disponível em: < http://www.cbeji.com.br/br/index.asp >. Acesso em: 15 de dez. 2004.

CHAN, Mark C.; GRIFFITH, Steven W.; IASI Anthony F. **Java – 1001 Dicas de Programação.** São Paulo: Makron Books, 1999

COUTINHO, S,C. **Números inteiros e criptografia RSA**. Rio de Janeiro: IMPA/SBM, 2000.

DINIZ, Davi Monteiro. Monteiro. **Documentos eletrônicos, assinaturas digitais: da qualificação jurídica dos arquivos digitais como documentos*.*** São Paulo: LTr, 1999.

GARFINKEL, Simson.**PGP: Pretty Good Privacy. O'Reilly & Associates.** São Paulo: Market Press,1995.

GARFINKEL, Simson; SPAFFORD, Gene.**Comércio e & Segurança na Web.** São Paulo, Market Press:1999.

JANDL, Peter Jr. **Introdução ao Java.** São Paulo, Berkeley:2002.

JANDL, Peter Jr. **Mais Java.** São Paulo, Futura:2003.

KAMINSKI, Omar.ICP-OAB - OAB-SP inicia a emissão dos certificados digitais. **Revista Consultor Jurídico, São Paulo, nov. 2002.** Disponível em: <http://conjur.uol.com.br/view.cfm>. Acesso em: 16 de dez. 2002.

KNUDSEN, Jonathan.**Java Cryptoghaphy.**Ed. O`reilly, 1998.

KUROSE, James F.; ROSS, Keith W. **Redes de Computadores: uma nova abordagem.** 1. ed. – São Paulo: Addison Wesley, 2003.

LUCCA, Newton De; SIMÃO FILHO, Adalberto. **Direito & Internet – aspectos jurídicos relevantes.** Bauru, SP:Edipro, 2000.

LYNCH, Daniel C., LUNDQUIST, Leslie. **Dinheiro Digital: o comércio na Internet.** Tradução por: Follow-up Traduções e Assessoria de Informática. Rio de Janeiro, Campus, 1996.

MARCACINI, A. T. Rosa. **Documento eletrônico como meio de prova.** <http://augustomarcacini.cjb.net/textos/docelet2.html>. Acesso em: 18 de Jul. 2003.

NEGROPONTE, Nicholas. **A Vida Digital**. São Paulo, SP: Companhia das Letras, 1995.

NORTHCUTT, Stephen ; ZELTSER, Lenny; WINTERS, Scott ; FREDERICK, Karen Kent ; RITCHEY, Ronald W. **Desvendando Segurança em Redes - O Guia Definitivo para Fortificação de Perímetros de Rede Usando Firewalls, Vpns, Roteadores e Sist.**, Rio de Janeiro:Campus, 2002.

OAKS, Scott. **Segurança de dados em Java.** Rio de Janeiro: Ciência Moderna, 1999.

SCHNEIER, Bruce. **Segurança . com – Segredos e mentiras sobre a proteção na vida digital**. Rio de Janeiro: Campus, 2001.

SCHNEIER, Bruce. **Applied Cryptography: protocols, algorithms, and source code in C –** 2°. ed – New Jersey,1996.

STALLINGS, William. **Cryptography and Network Security:** principles and practice – 2. ed – New Jersey: Prentice Hall,1999**.**

STOHLER,Paulo. **Criptografia: Conceitos Básicos.** Segunda Parte Future Technologies**,** fevereiro. 2002. Disponível em:

<http://www.fti.com.br/n_jornal/artigo_paulo_cripto02.htm>. Acesso em: 20 de mar. 2003.

SUN, Microsystem.**JavaTM Cryptography Extension (JCE) -** ***Reference Guide for the JavaTM 2 SDK, Standard Edition, v 1.4*****,** 2002. Disponívelem: <http://java.sun.com/j2se/1.4.1/docs/guide/security/jce/JCERefGuide.html>. Acesso em: 07 de dez. 2003.

SUN, Microsystem.**JavaTM Cryptography Architecture -** ***API Specification & Reference,***2002. Disponível em: <http://java.sun.com/j2se/1.4.1/docs/guide/security/CryptoSpec.html>. Acesso em: 07 de dez.2003.

SUN, Microsystem. **Java TM – 2 Plataform.** Disponível em: <http://java.sun.com/java2/whatis/>. Acesso em: 07 de dez. 2003.

SUN, Microsystem.**JavaTechnology – The Early Years,** 1998. Disponível em: <http://java.sun.com/features/1998/05/birthday.html>. Acesso em: 13 de Dez.2003.

TERADA, Routo. **Laboratório de Segurança de Dados,** 2003. Disponível em: <http://lsd.ime.usp.br/> Acesso em: 02 de Mar. 2004.

TERADA, Routo. **Segurança de Dados: Criptografia em Redes de Computador**. São Paulo: Edgard Blucher, 2000.

VOLPI, Marlon M.**Assinatura Digital: Aspectos Técnicos, Práticos e Legais**. Rio de Janeiro: Axcel Books do Brasil, 2001.

BIBLIOGRAFIA CONSULTADA

CAMPIONE, M; WALRATH, K. **The Java Tutorial: Object-OrientedProgramming for the Internet**. [S.l.]: SunSoft Press, 1996.

DEITEL, Harvet M.; DEITEL, Paul J. **Java como Programar**. 4ª Edição – Porto Alegre, 2003.

BRASIL, Banco. **Novo SPB – Sistema de Pagamentos Brasileiros**. 2002. Disponível em:<http://www.bcb.gov.br/htms/novaPaginaSPB/selic.asp>. Acesso em: 15 de Abr. 2004.

BARRETO, Paulo. **Escola Politécnica da Universidade de São Paulo,** 2003. Disponível em: <http://planeta.terra.com.br/informatica/paulobarreto/> Acesso em: 02 Mar. 2004.

CASTRO, Glauco César. **Assinatura Digital.** 2003. Disponível em: <http://www.mail-archive.com/java-list@soujava.org.br/msg37850.html> Acesso em: 04 de Mai. 2004.

DELLANI, Paulo Roberto. **Algoritmo de Encriptação MARS e AES.** Disponível em: < http://www.lisha.ufsc.br/~pdellani/cyphers/ > Acesso em 02 de Mai. 2004.

DIREITO na Web. **Aspectos Jurídicos Relacionados à Internet.** 2003. Disponível em: http://www.direitonaweb.com.br/. Acesso em: 18 de Mai. 2004.

FLEURY, A. Monclaro. **Gerência de Redes.** 1996. Disponível em: <http://www.penta.ufrgs.br/gere96/segur/cripto_.htm> Acesso em 13 de Set. 2004.

IACR International Association for Cryptologic Research. Disponível em: <http://www.iacr.org/~iacr>. Acesso em: 12 de Nov. 2003.

ITI – Instituto Nacional de Tecnologia da Informação. Disponível em: <http://www.iti.br/> Acesso em: 23 Jun. 2004.

KURNIAWAN, Budi. **Java para a Web com Servlets, JSP e EJB.** Rio de Janeiro: Ciência Moderna, 2002.

LOPES, Tarcisio. **Introdução ao Java,** 2001. Disponível em: <http://www.tarcisiolopes.com/intro_java/> Acesso em: 29 de Out. 2003.

MAIA, L. Paulo; PAGLIUSI, Sergio. **Criptografia e Cerificação.** Disponível em: <http://www.training.com.br/sic/pub_seg_cripto.htm>. Acesso em: 13 de Jun. 2004.

MARCACINI, A. T. Rosa. **Documento eletrônico como meio de prova.** <http://augustomarcacini.cjb.net/textos/docelet2.html>. Acesso em: 18 de Jul. 2003.

MENESES, Alfredo; OORSCHOT, Paul; VANSTONE, Scott. **Handbook of Applied Cryptography.** 1997. Disponível em: <http://www.cacr.math.uwaterloo.ca/hac>. Acesso em: 07 de Fev. 2004.

MICROSOFT Corporation. **Centro de Orientações de Segurança.** 2004. Disponível em: <http://www.microsoft.com/brasil/security/guidance/topics/devsec/secmod39.mspx>. Acesso em: 13 de Mai. 2004.

REZENDE, P. A. Dourado. **Departamento da Ciência da computação da Universidade de Brasília**. 2002. Disponível em: <http://www.cic.unb.br/docentes/pedro/trabs/leis.htm> Acesso em: 23 de Ago. 2003.

SSH Communications Security. **Cryptography A – Z.** Disponível em: <http://www.ssh.com/support/cryptography>. Acesso em: 12 de Nov. 2003.

TRINTA, F. A. Mota; MACEDO, Rodrigo Cavalcanti. , **Um Estudo Sobre Criptografia e Assinatura Digital,** 1998. Disponível em: <http://www.di.ufpe.br/~flash/ais98/cripto/criptografia.htm>. Acesso em 02 de Out. 2003.

WANNER, P. C. Herman. **Universidade Federal do Rio Grande do Sul.** 2001. Disponível em: <http://www.inf.ufrgs.br/procpar/disc/cmp167/trabalhos/sem2001-1/T2/herrmann/>. Acesso em: 21 de Ago. 2004.

ANEXOS

Anexo A – Programas Fonte do Pacote “Assinatura.jar”

Programas fonte desenvolvido utilizando a linguagem de programação Java, que incorporam o pacote de assinatura digital proposto contendo os principais recursos para efetuar a assinatura digital e a verificação do arquivo assinado, assim como a criação do par de chaves necessário.

- **Classe geraChaves.java**

```
//
package assinaturaDigital;

import java.io.*;
import java.security.*;
import com.sun.crypto.provider.SunJCE;

/** A classe <CODE>geraChaves</CODE> é onde gera as chaves Publica e
Privada
 * necessária para a Assinatura Digital.
 * @author Paulo Deliberador
 * @since 1.3
 * @version 1.0
 * @see assinaturaDigital.assinatura
 * @see assinaturaDigital.dadosAssinatura
 * @see assinaturaDigital.verificaAssinatura
 * @see verificaChaves
 */
public class geraChaves {
    private static boolean DEBUG = true;

    /** Inicializa um novo Objeto geraChaves */
    public geraChaves() {
        //
    }


    /** Gera o Par de Chaves para a Assinatura uma Chave Publica e outra
Privada.
     * @param arquivo - Nome do arquivo que será gravado as chaves
     * @param seed - "Semente" ou frase-senha para geração da chave
     * @return Retorna True caso tenha sido bem sucedida a geração do par
de chaves
     */
    private boolean gerarChaves(String arquivo, String seed) {
        boolean gera = false;
        try {
            SunJCE jce = new SunJCE();// criar instancia do provider
SunJCE. pode ser substituído por qualquer outro provider que implemente DSA
            Security.addProvider(jce);// adiciona SunJCE à lista de
providers
            // instancia objeto KeyPairGenerator para gerar par de chaves
do DSA
```

```
            KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA");
            // instancia objeto de geração de números pseudo-aleatorios
para utilização em SHA1 e DSA
            SecureRandom random = SecureRandom.getInstance("SHA1PRNG",
"SUN");
            // alimenta o gerador de números pseudo-aleatorios com a
semente digitada pelo usuário
            random.setSeed(seed.getBytes());
            keyGen.initialize(1024, random);// inicializa o gerador de
chaves com o número pseudo-aleatorio
            // gera par de chaves (publica e privada) a partir dos
parâmetros anteriores
            KeyPair parChaves = keyGen.generateKeyPair();
            // obtém do KeyPair as chaves publica e privada e armazena em
objetos adequados
            PublicKey pubKey = parChaves.getPublic();
            PrivateKey priKey = parChaves.getPrivate();
            // grava arquivos de chave serializando os objetos
            // o objeto serializado pode ser utilizado para armazenamento
em diversos modos, como por exemplo
            //   bancos de dados (campos BLOB), arquivos e outros
            ObjectOutputStream fpub = new ObjectOutputStream(new
FileOutputStream(arquivo+".pub"));
            ObjectOutputStream fpri = new ObjectOutputStream(new
FileOutputStream(arquivo+".pri"));
            fpub.writeObject(pubKey);
            fpri.writeObject(priKey);
            fpub.close();
            fpri.close();
            gera = true;
        } catch (Exception e1) {
            gera = false;
            System.out.println("Exceção: "+e1);
        }

        return(gera);
    }

    /** Gera o Par de Chaves para a Assinatura uma Chave Pública e outra
Privada
     * @param arquivo - Nome do arquivo que será gravado as chaves
     * @param seed - "Semente" ou frase-senha para geração da chave
     * @return Retorna True caso tenha sido bem sucedida a geração do par
de chaves
     */
    public boolean setGeraChaves(String arquivo, String seed){
        assinaturaDigital.geraChaves gerar = new
assinaturaDigital.geraChaves();
        boolean gera = gerar.gerarChaves(arquivo, seed);
        return(gera);
    }

    /** Gera o Par de Chaves para a Assinatura uma Chave Pública e outra
Privada
     * @param arquivo - Nome do arquivo que será gravado as chaves
     * @param seed - "Semente" ou frase-senha para geração da chave
     * @return Retorna True caso tenha sido bem sucedida a geração do par
de chaves
     */
    public boolean setGeraChaves(File arquivo, String seed){
```

```
        assinaturaDigital.geraChaves gerar = new
assinaturaDigital.geraChaves();
        boolean gera = gerar.gerarChaves(arquivo.toString(), seed);
        return(gera);
    }
}
```

- **Classe assinatura.java**

```
package assinaturaDigital;

import java.io.*;
import java.security.*;
import java.security.spec.*;
import java.util.*;
import java.net.*;


/** A classe <CODE>assinatura</CODE> é onde será realizada a Assinatura
Digital de
 * um determinado arquivo
 * @author Paulo Deliberador
 * @since 1.3
 * @version 1.0
 * @see assinaturaDigital.dadosAssinatura
 * @see assinaturaDigital.geraChaves
 * @see assinaturaDigital.verificaAssinatura
 * @see verificaChaves
 */
public class assinatura {
    private static boolean DEBUG = true;

    /** Inicializa um novo Objeto assinatura */
    public assinatura() {
        //
    }

    /** Assina Documento
     * @param arquivo - Nome do arquivo a ser assinado
     * @param arqPrivada - Nome do arquivo com a chave privada
     * @param dados - Dados gerais da Assinatura
     * @return Retorna True caso tenha sido bem sucedida a Assinatura
Digital
     */
    private boolean Assinar(String arquivo, String arqPrivada) {
//    assinaturaDigital.util.Topicos topAssina = new
assinaturaDigital.util.Topicos((new java.io.File(arquivo+".assinatura")));
        boolean assinatura = false;
        try {
            // abrir arquivo da chave privada para leitura
            ObjectInputStream keyIn = new ObjectInputStream(new
FileInputStream(arqPrivada));
            // ler objeto do arquivo (ObjectStream) e armazenar em objeto
da classe PrivateKey
            PrivateKey chavePrivada = (PrivateKey) keyIn.readObject();
            keyIn.close();// fechar arquivo
            FileInputStream fin = new FileInputStream(arquivo);// abertura
do arquivo a ser assinado
```

```
            byte[] buffer = new byte[8192];// declaração de buffer para
assinatura e tamanho
            int tamanho;
            Signature dsa = Signature.getInstance("SHA1withDSA");//
instancia objeto de assinatura digital (Signature) baseado em DSA com SHA1
            dsa.initSign(chavePrivada);// inicia o engine de assinatura com
a chave privada fornecida
            while ((tamanho = fin.read(buffer)) != -1) {// ler todo o
arquivo
                dsa.update(buffer, 0, tamanho);// armazena dados do arquivo
no engine de assinatura
            }
            byte[] sig = dsa.sign();// gera a assinatura digital, baseada
nos dados inseridos a partir do arquivo
            fin.close();

            FileOutputStream signFile = new
FileOutputStream(arquivo+".assinatura");// gravação do arquivo de
assinatura
            signFile.write(sig);
            DataOutputStream dos = new DataOutputStream(signFile);
            assinaturaDigital.dadosAssinatura dadosAss = new
assinaturaDigital.dadosAssinatura();
            String dados = dadosAss.getDados(); // pega a frase para gravar
os dados da assinatura.
            dos.writeBytes(dados);
//            dos.writeChars(dados);
            //signFile.write(dados.getBytes());
            dos.close();
            signFile.close();
            assinatura = true;
            if (DEBUG) System.out.println("Assinatura em arquivo:
"+arquivo+".assinatura");

        } catch (Exception e1) {
            assinatura = false;
            System.out.println("Erro: "+e1);
            e1.printStackTrace();
        }
        return(assinatura);
    }


    /** obtém um array de bytes referente à Assinatura Digital do arquivo
que está sendo assinado.
     * @param arquivo - Nome do arquivo a ser assinado.
     * @param arqPrivada - Nome do arquivo com a chave privada.
     * @return Retorna o array de bytes da Assinatura Digital.
     */
    public byte[] getAssinatura(String arquivo, String arqPrivada) {
        byte[] assinatura = new byte[8192];
        try {
            // abrir arquivo da chave privada para leitura
            ObjectInputStream keyIn = new ObjectInputStream(new
FileInputStream(arqPrivada));
            // ler objeto do arquivo (ObjectStream) e armazenar em objeto
da classe PrivateKey
            PrivateKey chavePrivada = (PrivateKey) keyIn.readObject();
            keyIn.close();// fechar arquivo
            FileInputStream fin = new FileInputStream(arquivo);// abertura
do arquivo a ser assinado
```

```
            byte[] buffer = new byte[8192];// declaração de buffer para
assinatura e tamanho
            int tamanho;
            Signature dsa = Signature.getInstance("SHA1withDSA");//
instancia objeto de assinatura digital (Signature) baseado em DSA com SHA1
            dsa.initSign(chavePrivada);// inicia o engine de assinatura com
a chave privada fornecida
            while ((tamanho = fin.read(buffer)) != -1) {// ler todo o
arquivo
                dsa.update(buffer, 0, tamanho);// armazena dados do arquivo
no engine de assinatura
            }
            assinatura = dsa.sign();// gera a assinatura digital, baseada
nos dados inseridos a partir do arquivo
            fin.close();
        } catch (Exception e1) {
            System.out.println("Erro: "+e1);
            e1.printStackTrace();
        }
        return(assinatura);
    }


    /** obtém um array de bytes referente à Assinatura Digital do arquivo
que está sendo assinado.
     * @param arquivo - Nome do arquivo a ser assinado.
     * @param arqPrivada - Nome do arquivo com a chave privada.
     * @return Retorna o array de bytes da Assinatura Digital.
     */
    public byte[] getAssinatura(File arquivo, File arqPrivada) {
        byte[] assinatura = new byte[8192];
        try {
            // abrir arquivo da chave privada para leitura
            ObjectInputStream keyIn = new ObjectInputStream(new
FileInputStream(arqPrivada));
            // ler objeto do arquivo (ObjectStream) e armazenar em objeto
da classe PrivateKey
            PrivateKey chavePrivada = (PrivateKey) keyIn.readObject();
            keyIn.close();// fechar arquivo
            FileInputStream fin = new FileInputStream(arquivo);// abertura
do arquivo a ser assinado
            byte[] buffer = new byte[8192];// declaração de buffer para
assinatura e tamanho
            int tamanho;
            Signature dsa = Signature.getInstance("SHA1withDSA");//
instancia objeto de assinatura digital (Signature) baseado em DSA com SHA1
            dsa.initSign(chavePrivada);// inicia o engine de assinatura com
a chave privada fornecida
            while ((tamanho = fin.read(buffer)) != -1) {// ler todo o
arquivo
                dsa.update(buffer, 0, tamanho);// armazena dados do arquivo
no engine de assinatura
            }
            assinatura = dsa.sign();// gera a assinatura digital, baseada
nos dados inseridos a partir do arquivo
            fin.close();
        } catch (Exception e1) {
            System.out.println("Erro: "+e1);
            e1.printStackTrace();
        }
        return(assinatura);
```

```
    }


    /** Assina um arquivo passando parâmetros para efetuar a Assinatura
Digital
     * @param arquivo - Nome do arquivo a ser assinado
     * @param arqPrivada - Nome do arquivo com a chave privada
     * @return Retorna True caso tenha sido bem sucedida a Assinatura
Digital
     */
    public boolean setAssinatura(String arquivo, String arqPrivada){
        assinaturaDigital.assinatura assinar = new
assinaturaDigital.assinatura();
        boolean assinatura = assinar.Assinar(arquivo, arqPrivada);
        return(assinatura);
    }

    /** Assina um arquivo passando parâmetros para efetuar a Assinatura
Digital
     * @param arquivo - Nome do arquivo a ser assinado
     * @param arqPrivada - Nome do arquivo com a chave privada
     * @return Retorna True caso tenha sido bem sucedida a Assinatura
Digital
     */
    public boolean setAssinatura(File arquivo, File arqPrivada){
        assinaturaDigital.assinatura assinar = new
assinaturaDigital.assinatura();
        boolean assinatura = assinar.Assinar(arquivo.toString(),
arqPrivada.toString());
        return(assinatura);
    }

}
```

- **Classe verificaChaves.java:**

```
//
package assinaturaDigital;

import java.io.*;
import java.security.*;
import com.sun.crypto.provider.SunJCE;
//import java.security.spec.*;


/** A classe <CODE>verificaChaves</CODE> é onde se compara a frase senha
digitada
 *  no processo de geração da chave Privada realizada pela classe
<I>geraChaves</I>
 *  com uma frase passada como parâmetro.
 * @author Paulo Deliberador
 * @since 1.3
 * @version 1.0
 * @see assinaturaDigital.assinatura
 * @see assinaturaDigital.dadosAssinatura
 * @see assinaturaDigital.geraChaves
 * @see assinaturaDigital.verificaAssinatura
 */
public class verificaChaves {
```

```
    private static boolean DEBUG = true;
    /** Inicializa um novo Objeto verificaChaves */
    public verificaChaves() {
        //
    }


    /** Verifica a frase-senha da chave privada recebendo como parâmetro os
itens da classe pública <I>setVerificaChaves</I>
     * @param arquivo - Nome de arquivo da chave privada
     * @param seed - Frase-senha para verificação
     * @return Retorna True caso a chave-senha passada como parâmetro seja
a mesma digitada na
     * gravação do arquivo da chave privada gerada pela classe
<I>geraChaves</I>
     */
    private boolean verificarChaves(String arquivo, String seed) {
        boolean gera = false;
        PrivateKey arq = getKeyFile(arquivo);
        PrivateKey frz = getKeySeed(seed);
        boolean retorno = arq.equals(frz);
        return(retorno);
    }


    private PrivateKey getKeyFile(String arquivo){
        Object aa = "";
        PrivateKey chavePrivada = null;
        assinaturaDigital.verificaChaves verificar = new
assinaturaDigital.verificaChaves();
        if (DEBUG) System.out.println("Arquivo= "+arquivo);
        try{
            ObjectInputStream keyIn = new ObjectInputStream(new
FileInputStream(arquivo));
            // ler objeto do arquivo (ObjectStream) e armazenar em objeto
da classe PrivateKey
            chavePrivada = (PrivateKey) keyIn.readObject();
            keyIn.close();// fechar arquivo
        } catch (Exception e1) {
            System.out.println("Erro: "+e1);
            e1.printStackTrace();
        }

        String chave = "retorno ";
        return(chavePrivada);
    }

    private PrivateKey getKeySeed(String seed){
        Object aa = "";
        PrivateKey chavePrivada = null;
        assinaturaDigital.verificaChaves verificar = new
assinaturaDigital.verificaChaves();
        if (DEBUG) System.out.println("seed= "+seed);
        try{
            SunJCE jce = new SunJCE();// criar instancia do provider
SunJCE. pode ser substituído por qualquer outro provider que implemente DSA
            Security.addProvider(jce);// adiciona SunJCE à lista de
providers
            // instancia objeto KeyPairGenerator para gerar par de chaves
do DSA
            KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA");
```

```
            // instancia objeto de geração de números pseudo-aleatorios
para utilização em SHA1 e DSA
            SecureRandom random = SecureRandom.getInstance("SHA1PRNG",
"SUN");
            // alimenta o gerador de números pseudo-aleatorios com a
semente digitada pelo usuário
            random.setSeed(seed.getBytes());
            keyGen.initialize(1024, random);// inicializa o gerador de
chaves com o numero pseudo-aleatorio
            // gera par de chaves (publica e privada) a partir dos
parâmetros anteriores
            KeyPair parChaves = keyGen.generateKeyPair();
            // obtém do KeyPair as chaves publica e privada e armazena em
objetos adequados
            chavePrivada = parChaves.getPrivate();


        } catch (Exception e1) {
            System.out.println("Erro: "+e1);
            e1.printStackTrace();
        }

        String chave = "retorno ";
        return(chavePrivada);
    }



    /** Verifica a frase-senha da chave privada
     * @param arquivo - Nome de arquivo da chave privada
     * @param seed - Frase-senha para verificação
     * @return Retorna True caso a chave-senha passada como parâmetro seja
a mesma digitada na
     * gravação do arquivo da chave privada gerada pela classe
<I>geraChaves</I>
     */
    public boolean setVerificaChaves(String arquivo, String seed){
        assinaturaDigital.verificaChaves verificar = new
assinaturaDigital.verificaChaves();
        boolean gera = verificar.verificarChaves(arquivo, seed);
        return(gera);
    }

    /** Verifica a frase-senha da chave privada
     * @param arquivo - Nome de arquivo da chave privada
     * @param seed - Frase-senha para verificação
     * @return Retorna True caso a chave-senha passada como parâmetro seja
a mesma digitada na
     * gravação do arquivo da chave privada gerada pela classe
<I>geraChaves</I>
     */
    public boolean setVerificaChaves(File arquivo, String seed){
        assinaturaDigital.verificaChaves verificar = new
assinaturaDigital.verificaChaves();
        boolean gera = verificar.verificarChaves(arquivo.toString(), seed);
        return(gera);
    }
}
```

- **Classe verificaAssinatura.java**

```
//
package assinaturaDigital;

import java.io.*;
import java.security.*;
import java.security.spec.*;

/** A classe <CODE>verificaAssinatura</CODE> é onde se faz a verificação da
 * Assinatura Digital.
 * @author Paulo Deliberador
 * @since 1.3
 * @version 1.0
 * @see assinaturaDigital.assinatura
 * @see assinaturaDigital.dadosAssinatura
 * @see assinaturaDigital.geraChaves
 * @see verificaChaves
 */
public class verificaAssinatura{
    private static boolean DEBUG = true;

    /** Inicializa um novo Objeto verificaAssinatura */
    public verificaAssinatura() {
        //
    }
    /** Classe que verifica se a Assinatura teve êxito.
     * @param arquivo - Nome do arquivo a ser verificado
     * @param arqPublica - Nome do arquivo com a chave publica
     * @return Retorna True caso a Verificação for bem Sucedida
     */
    private boolean verificarAssinatura(String arquivo, String arqPublica)
{
        boolean valido = false;
        try {
            // leitura do arquivo de assinatura
            FileInputStream signFile = new
FileInputStream(arquivo+".assinatura");
            ByteArrayOutputStream recepcao = new ByteArrayOutputStream();
            int dado = 0;
            while ((dado = signFile.read()) != -1) {
                recepcao.write(dado);
            }
            byte[] sig = recepcao.toByteArray();// armazenamento da
assinatura em array de bytes (sig)
            ObjectInputStream keyIn = new ObjectInputStream(new
FileInputStream(arqPublica));// leitura do arquivo de chave publica
            PublicKey chavePublica = (PublicKey) keyIn.readObject();//
armazena chave publica em objeto adequado (PublicKey)
            keyIn.close();
            FileInputStream fin = new FileInputStream(arquivo);// abertura
do arquivo a ser verificado
            byte[] buffer = new byte[8192];// declaração de buffer para
verificacao e tamanho
            int tamanho;
            Signature dsa = Signature.getInstance("SHA1withDSA");//
instancia do engine de verificacao
            dsa.initVerify(chavePublica);// inicializacao do engine de
verificacao com a chave publica
```

```
            while ((tamanho = fin.read(buffer)) != -1) {// ler arquivo a
ser verificado
                dsa.update(buffer, 0, tamanho);// armazenar dados no engine
de verificacao
            }
            // executar verificacao de assinatura a partir dos dados do
arquivo e da assinatura (sig)
            valido = dsa.verify(sig);
            fin.close();

        } catch (Exception e1) {
            System.out.println("Erro: Assinatura Invalida");
//            System.out.println("Erro: "+e1);
//            e1.printStackTrace();
        }

        return(valido);
    }


    /** Classe que verifica se a Assinatura teve êxito.
     * @param assinatura - Array de bytes contendo a assinatura digital
     * @param arquivo    - Nome do arquivo a ser verificado
     * @param arqPublica - Nome do arquivo com a chave publica
     * @return Retorna True caso a Verificação for bem Sucedida
     */
    private boolean verificarAssinaturaBanco(byte[] assinatura, String
arquivo, String arqPublica) {
        boolean valido = false;
        try {
            byte[] sig = assinatura;// armazenamento da assinatura em array
de bytes (sig)
            ObjectInputStream keyIn = new ObjectInputStream(new
FileInputStream(arqPublica));// leitura do arquivo de chave publica
            PublicKey chavePublica = (PublicKey) keyIn.readObject();//
armazena chave publica em objeto adequado (PublicKey)
            keyIn.close();
            FileInputStream fin = new FileInputStream(arquivo);// abertura
do arquivo a ser verificado
            byte[] buffer = new byte[8192];// declaração de buffer para
verificacao e tamanho
            int tamanho;
            Signature dsa = Signature.getInstance("SHA1withDSA");//
instancia do engine de verificacao
            dsa.initVerify(chavePublica);// inicializacao do engine de
verificacao com a chave publica
            while ((tamanho = fin.read(buffer)) != -1) {// ler arquivo a
ser verificado
                dsa.update(buffer, 0, tamanho);// armazenar dados no engine
de verificacao
            }
            // executar verificacao de assinatura a partir dos dados do
arquivo e da assinatura (sig)
            valido = dsa.verify(sig);
            fin.close();

        } catch (Exception e1) {
            System.out.println("Erro: Assinatura Invalida");
//            System.out.println("Erro: "+e1);
//            e1.printStackTrace();
        }
```

```
        return(valido);
    }


    /** Verifica a Assinatura Digital do arquivo Assinado.
     * @param arquivo - Nome do arquivo a ser verificado;
     * @param arqPublica - Nome do arquivo com a chave publica;
     * @return Retorna True caso a Verificação for bem Sucedida.
     */
    public boolean setVerificaAssinatura(String arquivo, String
arqPublica){
        assinaturaDigital.verificaAssinatura verificar = new
assinaturaDigital.verificaAssinatura();
        boolean ver = verificar.verificarAssinatura(arquivo, arqPublica);
        return(ver);
    }

    /** Verifica a Assinatura Digital do arquivo Assinado.
     * @param arquivo - Nome do arquivo a ser verificado;
     * @param arqPublica - Nome do arquivo com a chave publica;
     * @return Retorna True caso a Verificação for bem Sucedida.
     */
    public boolean setVerificaAssinatura(File arquivo, File arqPublica){
        assinaturaDigital.verificaAssinatura verificar = new
assinaturaDigital.verificaAssinatura();
        boolean ver = verificar.verificarAssinatura(arquivo.toString(),
arqPublica.toString());
        return(ver);
    }

    /** Verifica a Assinatura Digital do arquivo Assinado.
     * @param assinatura - Array de Bytes contendo os dados da Assinatura
Digital;
     * @param arquivo - Nome do arquivo a ser verificado;
     * @param arqPublica - Nome do arquivo com a chave publica;
     * @return Retorna True caso a Verificação for bem Sucedida.
     */
    public boolean setVerificaAssinatura(byte[] assinatura, File arquivo,
File arqPublica){
        assinaturaDigital.verificaAssinatura verificar = new
assinaturaDigital.verificaAssinatura();
        boolean ver = verificar.verificarAssinaturaBanco(assinatura,
arquivo.toString(), arqPublica.toString());
        return(ver);
    }

    /** Verifica a Assinatura Digital do arquivo Assinado.
     * @param assinatura - Array de Bytes contendo os dados da Assinatura
Digital;
     * @param arquivo - Nome do arquivo a ser verificado;
     * @param arqPublica - Nome do arquivo com a chave publica;
     * @return Retorna True caso a Verificação for bem Sucedida.
     */
    public boolean setVerificaAssinatura(byte[] assinatura, String arquivo,
String arqPublica){
        assinaturaDigital.verificaAssinatura verificar = new
assinaturaDigital.verificaAssinatura();
        boolean ver = verificar.verificarAssinaturaBanco(assinatura,
arquivo, arqPublica);
        return(ver);
```

```
    }

}
```

- **Classe dadosAssinatura.java**

```
//
package assinaturaDigital;

import java.io.*;
import java.util.*;
import java.net.*;
import java.text.*;

/** A classe <CODE>dadosAssinatura</CODE> é onde se define e/ou obtém os
dados da
 * Assinatura Digital assim como Nome do Autor, Data e Horas da Assinatura,
etc...
 * @author Paulo Deliberador
 * @since 1.3
 * @version 1.0
 * @see assinaturaDigital.assinatura
 * @see assinaturaDigital.geraChaves
 * @see assinaturaDigital.verificaAssinatura
 * @see verificaChaves
 */
public class dadosAssinatura {
    private static boolean DEBUG = false;
    private static String autorAssinatura = " ";
    private static String dataAssinatura = " ";
    private static String horaAssinatura = " ";
    /** Dados da Assinatura */
    private static File   pathAssinatura = new File(" ");// pathAssinatura
= " ";
    private static String ipAssinatura = " ";
    private static String maquinaAssinatura = "";
    private static String cabecalho = "\r\n\r\n";
//    private static String tkz = "|";
    private static boolean dados = false;

    /** Inicializa um novo Objeto dadosAssinatura */
    public dadosAssinatura() {
    }


    /** Inicializa um novo Objeto dadosAssinatura passando como parâmetro o
arquivo assinado desejado */
    public dadosAssinatura(String arquivo) {
        pathAssinatura = new File(arquivo+".assinatura");
    }


    /** Seta um Nome para o autor da Assinatura Digital
     * @param nAutor - Nome de quem esta assinando o arquivo
     */
    public void setAutor(String nAutor){
        assinaturaDigital.dadosAssinatura.autorAssinatura = nAutor;
        dados = true;
    }
```

```
    /** Seta a Data atual do sistema para a Assinatura Digital
     */
    public void setData(){
        Date hoje = new Date();
        String dataHora =  hoje.toLocaleString();
        StringTokenizer dataHoraTkz = new StringTokenizer(dataHora, " ");
        assinaturaDigital.dadosAssinatura.dataAssinatura =
dataHoraTkz.nextToken();
        dados = true;
    }

    /** Seta uma Data para a Assinatura Digital
     * @param nData - Data da Assinatura do arquivo (DD/MM/AAAA)
     */
    public void setData(String nData){
        Date hoje = new Date();
        assinaturaDigital.dadosAssinatura.dataAssinatura = nData;
        dados = true;
    }

    /** Seta a Hora atual do Sistema para a Assinatura Digital
     */
    public void setHora(){
        Date hoje = new Date();
        String dataHora =  hoje.toLocaleString();
        StringTokenizer dataHoraTkz = new StringTokenizer(dataHora, " ");
        dataHoraTkz.nextToken();
        assinaturaDigital.dadosAssinatura.horaAssinatura =
dataHoraTkz.nextToken();
        dados = true;
    }

    /** Seta uma Hora para a Assinatura Digital
     * @param nHora - Horas da Assinatura do arquivo (HH:MM)
     */
    public void setHora(String nHora){
        assinaturaDigital.dadosAssinatura.horaAssinatura = nHora;
        dados = true;
    }

    /** Seta o Nome da Maquina onde o Arquivo está sendo Assinado
     */
    public void setMaquina(){
        try{
            assinaturaDigital.dadosAssinatura.maquinaAssinatura =
InetAddress.getLocalHost().getHostName();
            dados = true;
        } catch (UnknownHostException e) {
            System.err.println("Endereço não Encontrado: " + e);
        }
    }

    /** Seta o Diretório(caminho) em que o arquivo assinado esta localizado
mais o nome do arquivo.
     * @param nPath Diretório(caminho) mais nome do arquivo que está sendo
assinado
     */
    public void setPath(String nPath){
        assinaturaDigital.dadosAssinatura.pathAssinatura = new File(nPath);
    }
```

```
    /** Seta o endereço IP da Maquina onde o Arquivo está sendo Assinado
     */
    public void setIp(){
        try{
            assinaturaDigital.dadosAssinatura.ipAssinatura =
InetAddress.getLocalHost().getHostAddress();
            dados = true;
        } catch (UnknownHostException e) {
            System.err.println("Endereço não Encontrado: " + e);
        }
    }

    /** Seta o Nome de Quem Esta Assinando o Arquivo, a Data e a Hora de
Assinatura.
     * @param nAutor - Nome de quem esta assinando o arquivo
     */
    public void setDefault(String nAutor){
        assinaturaDigital.dadosAssinatura.autorAssinatura = nAutor;
        setData();
        setHora();
        dados = true;
    }

    /** obtém uma frase contendo todos os dados embaralhados setados para a
     * Assinatura.
     * @return Retorna os Dados da Assinatura separados pelo caracter |.
     */
    public String getDados(){
        String frase = "NADA";
        if(dados){
            frase = autorAssinatura +"|"+ dataAssinatura +"|"+
horaAssinatura +"|"+
            pathAssinatura +"|"+ maquinaAssinatura +"|"+ ipAssinatura;
        }
        String fraseMudada = charToBytes(frase.getBytes());
        return(cabecalho+"["+fraseMudada+"]");
    }

    /** obtém uma frase contendo todos os dados sem alteração para a
Assinatura.
     * @return Retorna os Dados da Assinatura separados pelo caracter |.
     */
    public String getDadosBanco(){
        String frase = "NADA";
        if(dados){
            frase = autorAssinatura +"|"+ dataAssinatura +"|"+
horaAssinatura +"|"+
            pathAssinatura +"|"+ maquinaAssinatura +"|"+ ipAssinatura;
        }
        return(frase);
    }

    /** Pega a frase completa contendo os dados da assinatura
     * @param arquivo Arquivo Assinado
     * @return A Frase Completa
     */
    private String getDadosAssinado(File arquivo){
//        File arquivo = arq;
        String frase = "";
```

```
        try{
            if (arquivo.exists() && arquivo.canRead()) {
                FileInputStream istream = new FileInputStream(arquivo);
                BufferedReader BuffIn = new BufferedReader(new
InputStreamReader(istream));
                while (BuffIn.ready()) {
                    String tmpRead = BuffIn.readLine().trim();
                    if ((tmpRead.startsWith("[")) &&
(tmpRead.endsWith("]"))) {
                        frase = tmpRead.substring(1, tmpRead.length() -
1).trim();
                        continue;
                    }
                }
                BuffIn.close();
                istream.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return(frase);
    }
    /** Pega a Frase contendo no Arquivo de Assinatura que contenha os
dados da Assinatura
     * @param arquivo Nome do Arquivo que foi Assinado
     * @return Retorna a Frase Decriptografada
     */
    private String getFrase(){
            StringBuffer frase = new StringBuffer();
            String fraseAscii = this.getDadosAssinado(pathAssinatura);
            if(!fraseAscii.trim().equalsIgnoreCase("NADA")){
                StringTokenizer fraseTkn = new StringTokenizer(fraseAscii,
"^");
                while(fraseTkn.hasMoreTokens()){
                    int ascii = Integer.parseInt(fraseTkn.nextToken()); //
pega cada campo da frase já convertendo em inteiro
                    String conversao = ""+((ascii/2)-4); // Faz a conta
para voltar o numero ao normal e tranforma em uma String.
                    char caracter =
(char)Integer.valueOf(conversao).intValue(); //Cria um caracter da String
                    frase.append(caracter);//acrescenta uma letra na frase
                }
            }
            return(frase.toString());//Retorna a Frase completa.
    }

    /** obtém a Nome de quem Assinou o Arquivo.
     * @return Retorna o Nome de quem Assinou o Arquivo.
     * @see assinaturaDigital.dadosAssinatura#setPath(java.lang.String)
     * É necessário utilizar o comando <b>setPath</b> caso não tenha
instanciado a classe
     * <B>dadosAssinatura</B> sem passar o parâmetro do caminho do arquivo;
     */
    public String getAutor(){
        String nome = "";
        //        String frase = this.getDadosAssinado(arquivo);
        String frase = this.getFrase();
        if(!frase.trim().equalsIgnoreCase("NADA")){
            StringTokenizer fraseTkn = new StringTokenizer(frase, "|");
            nome = fraseTkn.nextToken();
        }
```

```
        return(nome);
    }
    /** obtém a Data que o arquivo foi assinado
     * @return Retorna a Data que o arquivo foi assinado
     * @see assinaturaDigital.dadosAssinatura#setPath(java.lang.String)
     * É necessário utilizar o comando <b>setPath</b> caso não tenha
instanciado a classe
     * <B>dadosAssinatura</B> sem passar o parâmetro do caminho do arquivo;
     */
    public String getData(){
        String data = "";
        //        String frase = this.getDadosAssinado(arquivo);
        String frase = this.getFrase();
        if(!frase.trim().equalsIgnoreCase("NADA")){
            StringTokenizer fraseTkn = new StringTokenizer(frase, "|");
            fraseTkn.nextToken();
            data = fraseTkn.nextToken();
        }
        return(data);
    }
    /** obtém a Hora que o arquivo foi assinado
     * @return Retorna a Hora que o arquivo foi assinado
     * @see assinaturaDigital.dadosAssinatura#setPath(java.lang.String)
     * É necessário utilizar o comando <b>setPath</b> caso não tenha
instanciado a classe
     * <B>dadosAssinatura</B> sem passar o parâmetro do caminho do arquivo;
     */
    public String getHora(){
        String horas = "";
        //        String frase = this.getDadosAssinado(arquivo);
        String frase = this.getFrase();
        if(!frase.trim().equalsIgnoreCase("NADA")){
            StringTokenizer fraseTkn = new StringTokenizer(frase, "|");
            fraseTkn.nextToken();
            fraseTkn.nextToken();
            horas = fraseTkn.nextToken();
        }
        return(horas);
    }
    /** obtém o Diretório(caminho) e o nome do Arquivo que foi Assinado.
     * @return Retorna o Diretório(caminho) e o nome do Arquivo que foi
Assinado.
     * @see assinaturaDigital.dadosAssinatura#setPath(java.lang.String)
     * É necessário utilizar o comando <b>setPath</b> caso não tenha
instanciado a classe
     * <B>dadosAssinatura</B> sem passar o parâmetro do caminho do arquivo;
     */
    public String getPath(){
        String path = "";
        //        String frase = this.getDadosAssinado(arquivo);
        String frase = this.getFrase();
        if(!frase.trim().equalsIgnoreCase("NADA")){
            StringTokenizer fraseTkn = new StringTokenizer(frase, "|");
            fraseTkn.nextToken();
            fraseTkn.nextToken();
            fraseTkn.nextToken();
            path = fraseTkn.nextToken();
        }
        return(path);
    }
```

```
    /** obtém o Nome do Computador que Assinou o Arquivo.
     * @return Retorna Nome do Computador que Assinou o Arquivo.
     * @see assinaturaDigital.dadosAssinatura#setPath(java.lang.String)
     * É necessário utilizar o comando <b>setPath</b> caso não tenha
instanciado a classe
     * <B>dadosAssinatura</B> sem passar o parâmetro do caminho do arquivo;
     */
    public String getMaquina(){
        String maquina = "";
        //        String frase = this.getDadosAssinado(arquivo);
        String frase = this.getFrase();
        if(!frase.trim().equalsIgnoreCase("NADA")){
            StringTokenizer fraseTkn = new StringTokenizer(frase, "|");
            fraseTkn.nextToken();
            fraseTkn.nextToken();
            fraseTkn.nextToken();
            fraseTkn.nextToken();
            maquina = fraseTkn.nextToken();
        }
        return(maquina);
    }
    /** obtém o Endereço de Ip do Computador que Assinou o Arquivo.
     * @return Retorna Endereço de Ip do Computador que Assinou o Arquivo.
     * @see assinaturaDigital.dadosAssinatura#setPath(java.lang.String)
     * É necessário utilizar o comando <b>setPath</b> caso não tenha
instanciado a classe
     * <B>dadosAssinatura</B> sem passar o parâmetro do caminho do arquivo;
     */
    public String getIp(){
        String ip = "";
        //        String frase = this.getDadosAssinado(arquivo);
        String frase = this.getFrase();
        if(!frase.trim().equalsIgnoreCase("NADA")){
            StringTokenizer fraseTkn = new StringTokenizer(frase, "|");
            fraseTkn.nextToken();
            fraseTkn.nextToken();
            fraseTkn.nextToken();
            fraseTkn.nextToken();
            fraseTkn.nextToken();
            ip = fraseTkn.nextToken();
        }
        return(ip);
    }
    /** Tranforma um Array de Bytes em código ASCII + 4 * 2.
     * @param byteArray Frase para alteração no formato byteArray
     * @return Uma String contendo a frase alterada
     */
    private static String charToBytes(byte byteArray[]) {
        StringBuffer strBuff1   =   new StringBuffer();
        for (int i = 0; i < byteArray.length; i++) {
            int numTemp = byteArray[i];
            int num   = (numTemp+4)*2;
            if(DEBUG) System.out.println("num= "+num);
            strBuff1.append(num+"^");
        }
        return strBuff1.toString();
    }

}
```

Anexo B – Programas fonte utilizando o pacote “Assinatura.jar”

Programas fonte desenvolvido utilizando a linguagem de programação Java, contendo alguns exemplos de utilização do pacote de assinatura digital.

- **Classe Fr_Assinatura.java**

```
/*
 * assinaturaDigital.java
 *
 * Created on 20 de Outubro de 2003, 16:28
 */


// Fazer com que o usuário digite a mesma frase gerada na chave privada para que esta seja validada na hora de assinar o documento (OK)
// Colocar dados como nome do usuário e data da Assinatura no arquivo de .assinatura (OK)
// Verificar e emitir msg quando assinar pelo banco
// Auto acrescenta o código da tabela quando for incluir


package Assinatura.controle;
import java.io.*;
import javax.swing.*;

import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.util.*;
//import assinaturaDigital.*;

/**
 *
 * @author  Paulo Deliberador
 */
public class Fr_Assinatura extends javax.swing.JFrame {
    private static boolean DEBUG = true;

    private javax.swing.JFileChooser jfcArquivo = new javax.swing.JFileChooser();
    public Object[] options = { "OK" }; // botão mensagem dos frames de avisos
    public Object[] btSimNao = { "Sim","Não" }; // botões mensagem dos frames de avisos

    public static String caminho;

    static {
        if ( System.getProperty("CAMINHO") == null ) {
            caminho = System.getProperty("user.dir");
```

```
            java.util.StringTokenizer           st           =           new
java.util.StringTokenizer(System.getProperty("java.class.path"),
java.io.File.pathSeparator);
            while (st.hasMoreTokens()){
                String tkn = st.nextToken();
                if ( tkn.toUpperCase().endsWith("ASSINATURA.JAR") ) {
                    int pos = tkn.lastIndexOf(java.io.File.separator);
                    caminho  =  (  pos  >=  0  ?  tkn.substring(0,  pos)  :
System.getProperty("user.dir") );
                    break;
                }
            }
        } else {
            caminho = System.getProperty("CAMINHO");
        }
        System.out.println(Fr_Assinatura.caminho);
    }


    /** Creates new form assinaturaDigital */
    public Fr_Assinatura() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {
        jPanel1 = new javax.swing.JPanel();
        jbGeraChaves = new javax.swing.JButton();
        jbAssinar = new javax.swing.JButton();
        jbVerificar = new javax.swing.JButton();
        jbDados = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jtfArquivo = new javax.swing.JTextField();
        jbArquivo = new javax.swing.JButton();
        jtfAutor = new javax.swing.JTextField();
        jLabel2 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jcbBanco = new javax.swing.JCheckBox();

        getContentPane().setLayout(new
org.netbeans.lib.awtextra.AbsoluteLayout());

        setTitle("Assinatura Digital");
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });

        jPanel1.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        jPanel1.setBorder(new javax.swing.border.EtchedBorder());
        jbGeraChaves.setText("Gerar Chaves");
        jbGeraChaves.addActionListener(new  java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jbGeraChavesActionPerformed(evt);
```

```
            }
        });

        jPanel1.add(jbGeraChaves,                                          new
org.netbeans.lib.awtextra.AbsoluteConstraints(20, 90, -1, -1));

        jbAssinar.setText("Assinar");
        jbAssinar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jbAssinarActionPerformed(evt);
            }
        });

        jPanel1.add(jbAssinar,                                             new
org.netbeans.lib.awtextra.AbsoluteConstraints(130, 90, -1, -1));

        jbVerificar.setText("Verificar");
        jbVerificar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jbVerificarActionPerformed(evt);
            }
        });

        jPanel1.add(jbVerificar,                                           new
org.netbeans.lib.awtextra.AbsoluteConstraints(210, 90, -1, -1));

        jbDados.setText("Dados");
        jbDados.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jbDadosActionPerformed(evt);
            }
        });

        jPanel1.add(jbDados,                                               new
org.netbeans.lib.awtextra.AbsoluteConstraints(290, 90, -1, -1));

        jLabel1.setText("Arquivo:");
        jPanel1.add(jLabel1,                                               new
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 40, -1, -1));

        jtfArquivo.setText(" ");
        jPanel1.add(jtfArquivo,                                            new
org.netbeans.lib.awtextra.AbsoluteConstraints(80, 40, 400, -1));

        jbArquivo.setText("...");
        jbArquivo.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jbArquivoActionPerformed(evt);
            }
        });

        jPanel1.add(jbArquivo,                                             new
org.netbeans.lib.awtextra.AbsoluteConstraints(490, 40, 60, 20));

        jtfAutor.setText(" ");
        jPanel1.add(jtfAutor,                                              new
org.netbeans.lib.awtextra.AbsoluteConstraints(80, 10, 400, -1));

        jLabel2.setText("Assinante:");
        jPanel1.add(jLabel2,                                               new
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 10, -1, -1));
```

```
        jButton1.setText("Fechar");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jPanel1.add(jButton1,                                                new
org.netbeans.lib.awtextra.AbsoluteConstraints(480, 90, -1, -1));

        jcbBanco.setText("Banco");
        jPanel1.add(jcbBanco,                                                new
org.netbeans.lib.awtextra.AbsoluteConstraints(490, 10, -1, -1));

        getContentPane().add(jPanel1,                                        new
org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 560, 130));

        pack();
    }

    private void jbDadosActionPerformed(java.awt.event.ActionEvent evt) {
        // Mostra os dados Da Assinatura
        if(jcbBanco.isSelected()){
            dadosBanco();
        }else{
            dadosArquivo();
        }

    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // Add your handling code here:
        System.exit(0);
    }

    private void jbArquivoActionPerformed(java.awt.event.ActionEvent evt) {
        // Add your handling code here:
        Assinatura.util.ExampleFileFilter        filter        =        new
Assinatura.util.ExampleFileFilter();
        filter.addExtension("txt");
        filter.addExtension("doc");
        filter.setDescription("Documentos de Texto");
        jfcArquivo.setFileFilter(filter);
        jfcArquivo.setCurrentDirectory(new
java.io.File(Fr_Assinatura.caminho));
        jfcArquivo.setDialogType(javax.swing.JFileChooser.OPEN_DIALOG);
        int retValor = jfcArquivo.showOpenDialog(this);
        if(retValor == javax.swing.JFileChooser.APPROVE_OPTION){
            try {
                File arquivo = jfcArquivo.getSelectedFile();
                jtfArquivo.setText(arquivo.toString());
            } catch (Exception e) {
                //LOG
            }
        }

    }

    private void jbVerificarActionPerformed(java.awt.event.ActionEvent evt)
{
```

```
        // Verifica se o Arquivo esta assinado
        if(jcbBanco.isSelected()){
            verificarBanco();
        }else{
            verificarArquivo();
        }

    }

    private void jbAssinarActionPerformed(java.awt.event.ActionEvent evt) {
        // Add your handling code here:
        if(jcbBanco.isSelected()){
            assinarBanco();
        }else{
           assinarArquivo();
        }
    }

    private   void   jbGeraChavesActionPerformed(java.awt.event.ActionEvent
evt) {
        // Add your handling code here:
        Assinatura.util.ExampleFileFilter        filter        =        new
Assinatura.util.ExampleFileFilter();
        assinaturaDigital.geraChaves          gerar          =          new
assinaturaDigital.geraChaves();
        String frase = JOptionPane.showInputDialog(this, "Entre com a Frase
Secreta",      "Frase      Secreta      para      Assinatura      Digital",
JOptionPane.QUESTION_MESSAGE);
        if(frase == null){
            frase = "";
        }
        if(!frase.trim().equalsIgnoreCase("")){
            filter.addExtension("pri");
            filter.addExtension("pub");
            filter.setDescription("Documentos de chave Publica e Privada");
            jfcArquivo.setFileFilter(filter);
            jfcArquivo.setCurrentDirectory(new
java.io.File(Fr_Assinatura.caminho));
            jfcArquivo.setDialogType(javax.swing.JFileChooser.SAVE_DIALOG);
            int retValor = jfcArquivo.showSaveDialog(this);
            if(retValor == javax.swing.JFileChooser.APPROVE_OPTION) {
                try {
                    File arquivo = jfcArquivo.getSelectedFile();
                    boolean  gera  =  gerar.setGeraChaves(arquivo.toString(),
frase);
                    if(gera){
                        JOptionPane.showOptionDialog(this,  "Chaves  gerada
com sucesso:\nChave Publica= "+
                        arquivo.toString()+".pub\nChave            Privada=
"+arquivo.toString()+".pri",
                        "Aviso",                JOptionPane.DEFAULT_OPTION,
JOptionPane.INFORMATION_MESSAGE,null, options, options[0]);
                    }else{
                        JOptionPane.showOptionDialog(this,     "Não     foi
possível     gerar     Chaves     Publica     e     Privada.",     "Erro",
JOptionPane.DEFAULT_OPTION,     JOptionPane.ERROR_MESSAGE,null,     options,
options[0]);
                    }
                } catch (Exception e) {
                    //LOG
                }
```

```
            }
        }else{
            JOptionPane.showOptionDialog(this, "Não foi Digitado uma Frase
Secreta.",               "Erro",               JOptionPane.DEFAULT_OPTION,
JOptionPane.ERROR_MESSAGE,null, options, options[0]);
        }
    }

    /** Assina o documento digitalmente gravando a assinatura em arquivo
.assinatura */
    private void assinarArquivo(){
        String arquivo = jtfArquivo.getText();
        boolean assinatura = false;
        Assinatura.util.ExampleFileFilter        filter        =        new
Assinatura.util.ExampleFileFilter();
        assinaturaDigital.assinatura                    assinar    =    new
assinaturaDigital.assinatura();
        assinaturaDigital.dadosAssinatura            dados      =      new
assinaturaDigital.dadosAssinatura();
        assinaturaDigital.verificaChaves                   key    =    new
assinaturaDigital.verificaChaves();

        dados.setAutor(jtfAutor.getText());
        dados.setData();    dados.setHora();
        dados.setIp();      dados.setMaquina();
        dados.setPath(arquivo);
        File arqPrivada = new File("");
        if(arquivo == null) arquivo = "";
        if(!arquivo.trim().equalsIgnoreCase("")){
            filter.addExtension("pri");
            filter.setDescription("Chave Privada");
            jfcArquivo.setFileFilter(filter);
            jfcArquivo.setCurrentDirectory(new
java.io.File(Fr_Assinatura.caminho));
            jfcArquivo.setDialogType(javax.swing.JFileChooser.OPEN_DIALOG);
            int retValor = jfcArquivo.showOpenDialog(this);
            if(retValor == javax.swing.JFileChooser.APPROVE_OPTION) {
                try {
                    arqPrivada = jfcArquivo.getSelectedFile();
                } catch (Exception e) {
                    //LOG
                }
            }else{
                JOptionPane.showOptionDialog(this,  "Não  foi  Especificado
uma      chave      Privada.",      "Erro",      JOptionPane.DEFAULT_OPTION,
JOptionPane.ERROR_MESSAGE,null, options, options[0]);
            }
            String  frase  =  JOptionPane.showInputDialog(this,  "Entre  com  a
Frase   Secreta",   "Confirme   a   Chave   Secreta   para   Assinatura",
JOptionPane.QUESTION_MESSAGE);
            boolean  confere  =  key.setVerificaChaves(arqPrivada.toString(),
frase);
            if(confere){
                assinatura         =         assinar.setAssinatura(arquivo,
arqPrivada.toString());
            }else{
                JOptionPane.showOptionDialog(this,   "Frase   Secreta   não
Confere\nNão     foi     Possível     Assinar     o     Arquivo",     "Erro",
JOptionPane.DEFAULT_OPTION,     JOptionPane.ERROR_MESSAGE,null,     options,
options[0]);
            }
```

```
            if(assinatura){
                JOptionPane.showOptionDialog(this, "Arquivo Assinado com
Sucesso",                "Aviso",                JOptionPane.DEFAULT_OPTION,
JOptionPane.INFORMATION_MESSAGE,null, options, options[0]);
            }else{
                JOptionPane.showOptionDialog(this,    "Problemas    com    a
Assinatura.\nNão    foi    Possível    Assinar    o    Arquivo",    "Erro",
JOptionPane.DEFAULT_OPTION,     JOptionPane.ERROR_MESSAGE,null,     options,
options[0]);
            }
        }else{
            JOptionPane.showOptionDialog(this,  "Não  foi  Especificado  um
Arquivo    para    Assinatura.",    "Erro",    JOptionPane.DEFAULT_OPTION,
JOptionPane.ERROR_MESSAGE,null, options, options[0]);
        }
    }

    /** Assina o documento digitalmente gravando a assinatura em Baco de
Dados */
    private void assinarBanco(){
        Assinatura.util.ExampleFileFilter            filter      =      new
Assinatura.util.ExampleFileFilter();
        assinaturaDigital.assinatura                    assinar    =    new
assinaturaDigital.assinatura();
        assinaturaDigital.dadosAssinatura               dados     =     new
assinaturaDigital.dadosAssinatura();
        assinaturaDigital.verificaChaves                     key    =    new
assinaturaDigital.verificaChaves();
        String arquivo = jtfArquivo.getText();
        String assinatura ="";

        dados.setAutor(jtfAutor.getText());
        dados.setData();    dados.setHora();
        dados.setIp();      dados.setMaquina();
        dados.setPath(arquivo);
        File arqPrivada = new File("");
        if(arquivo == null) arquivo = "";
        if(!arquivo.trim().equalsIgnoreCase("")){
            filter.addExtension("pri");
            filter.setDescription("Chave Privada");
            jfcArquivo.setFileFilter(filter);
            jfcArquivo.setCurrentDirectory(new
java.io.File(Fr_Assinatura.caminho));
            jfcArquivo.setDialogType(javax.swing.JFileChooser.OPEN_DIALOG);
            int retValor = jfcArquivo.showOpenDialog(this);
            if(retValor == javax.swing.JFileChooser.APPROVE_OPTION) {
                try {
                    arqPrivada = jfcArquivo.getSelectedFile();
                } catch (Exception e) {
                    //LOG
                }
            }else{
                JOptionPane.showOptionDialog(this,  "Não  foi  Especificado
uma      chave      Privada.",      "Erro",      JOptionPane.DEFAULT_OPTION,
JOptionPane.ERROR_MESSAGE,null, options, options[0]);
            }
            String  frase  =  JOptionPane.showInputDialog(this,  "Entre  com  a
Frase    Secreta",    "Confirme    a    Chave    Secreta    para    Assinatura",
JOptionPane.QUESTION_MESSAGE);
            boolean  confere  =  key.setVerificaChaves(arqPrivada.toString(),
frase);
```

```
            if(confere){
                byte[]      ass      =      assinar.getAssinatura(arquivo,
arqPrivada.toString());
                if(DEBUG)      System.out.println("Tamanho      Assinatura=
"+ass.length);
                for(int i=0;i<=ass.length-1;i++){
                    assinatura = assinatura + ass[i]+"^";
//                    assinatura = assinatura + (char)ass[i];
                }
                if(DEBUG) System.out.println("Assinatura= "+assinatura);
                Assinatura.util.assinaBanco        banco        =        new
Assinatura.util.assinaBanco(true);
                banco.verificaTabela(jtfArquivo.getText(),      assinatura,
dados.getDadosBanco());
                banco.shutDown();
            }else{
                JOptionPane.showOptionDialog(this,   "Frase   Secreta   não
Confere\nNão     foi     Possível     Assinar     o     Arquivo",     "Erro",
JOptionPane.DEFAULT_OPTION,     JOptionPane.ERROR_MESSAGE,null,     options,
options[0]);
            }
        }else{
            JOptionPane.showOptionDialog(this,  "Não  foi  Especificado  um
Arquivo    para    Assinatura.",    "Erro",    JOptionPane.DEFAULT_OPTION,
JOptionPane.ERROR_MESSAGE,null, options, options[0]);
        }

    }


    /** Verifica o documento assinado digitalmente conferindo a assinatura
no arquivo .assinatura */
    private void verificarArquivo(){
        Assinatura.util.ExampleFileFilter                   filter   =   new
Assinatura.util.ExampleFileFilter();
        assinaturaDigital.verificaAssinatura      verificar      =      new
assinaturaDigital.verificaAssinatura();
        String arquivo = jtfArquivo.getText();
        File arqPublica = new File("");
        if(!arquivo.trim().equalsIgnoreCase("")){
            filter.addExtension("pub");
            filter.setDescription("Chave Publica");
            jfcArquivo.setFileFilter(filter);
            jfcArquivo.setCurrentDirectory(new
java.io.File(Fr_Assinatura.caminho));
            jfcArquivo.setDialogType(javax.swing.JFileChooser.OPEN_DIALOG);
            int retValor = jfcArquivo.showOpenDialog(this);
            if(retValor == javax.swing.JFileChooser.APPROVE_OPTION) {
                try {
                    arqPublica = jfcArquivo.getSelectedFile();
                } catch (Exception e) {
                    //LOG
                }
            }else{
                JOptionPane.showOptionDialog(this,  "Não  foi  Especificado
uma      chave      Privada.",      "Erro",      JOptionPane.DEFAULT_OPTION,
JOptionPane.ERROR_MESSAGE,null, options, options[0]);
            }
            boolean     ok     =     verificar.setVerificaAssinatura(arquivo,
arqPublica.toString());
            if(ok){
```

```
                JOptionPane.showOptionDialog(this,   "Assinatura   Válida.",
"Aviso",  JOptionPane.DEFAULT_OPTION,  JOptionPane.INFORMATION_MESSAGE,null,
options, options[0]);
            }else{
                JOptionPane.showOptionDialog(this,  "Assinatura  Inválida",
"Aviso",     JOptionPane.DEFAULT_OPTION,     JOptionPane.ERROR_MESSAGE,null,
options, options[0]);
            }
        }else{
            JOptionPane.showOptionDialog(this,  "Não  foi  Especificado  um
Arquivo    para    Assinatura.",    "Erro",    JOptionPane.DEFAULT_OPTION,
JOptionPane.ERROR_MESSAGE,null, options, options[0]);
        }
    }

    /** Verifica o documento assinado digitalmente conferindo a assinatura
no Banco de Dados */
    private void verificarBanco(){
        Assinatura.util.ExampleFileFilter        filter        =        new
Assinatura.util.ExampleFileFilter();
        assinaturaDigital.verificaAssinatura      verificar      =      new
assinaturaDigital.verificaAssinatura();
        String arquivo = jtfArquivo.getText();
        File arqPublica = new File("");
        if(!arquivo.trim().equalsIgnoreCase("")){
            filter.addExtension("pub");
            filter.setDescription("Chave Publica");
            jfcArquivo.setFileFilter(filter);
            jfcArquivo.setCurrentDirectory(new
java.io.File(Fr_Assinatura.caminho));
            jfcArquivo.setDialogType(javax.swing.JFileChooser.OPEN_DIALOG);
            int retValor = jfcArquivo.showOpenDialog(this);
            if(retValor == javax.swing.JFileChooser.APPROVE_OPTION) {
                try {
                    arqPublica = jfcArquivo.getSelectedFile();
                } catch (Exception e) {
                    //LOG
                }
            }else{
                JOptionPane.showOptionDialog(this,  "Não  foi  Especificado
uma      chave      Privada.",      "Erro",      JOptionPane.DEFAULT_OPTION,
JOptionPane.ERROR_MESSAGE,null, options, options[0]);
            }
            Assinatura.util.assinaBanco         dados         =         new
Assinatura.util.assinaBanco(true);
            byte[] assByte = dados.getAssinaturaBanco(arquivo);
            boolean    ok    =    verificar.setVerificaAssinatura(assByte,
arquivo.toString(), arqPublica.toString());
            dados.shutDown();
            if(ok){
                JOptionPane.showOptionDialog(this,   "Assinatura   Válida.",
"Aviso",  JOptionPane.DEFAULT_OPTION,  JOptionPane.INFORMATION_MESSAGE,null,
options, options[0]);
            }else{
                JOptionPane.showOptionDialog(this,  "Assinatura  Inválida",
"Aviso",     JOptionPane.DEFAULT_OPTION,     JOptionPane.ERROR_MESSAGE,null,
options, options[0]);
            }
        }else{
```

```
            JOptionPane.showOptionDialog(this, "Não foi Especificado um
Arquivo    para    Assinatura.",    "Erro",    JOptionPane.DEFAULT_OPTION,
JOptionPane.ERROR_MESSAGE,null, options, options[0]);
        }
    }

    /** Obtem os dados da assinatura do documento assinado digitalmente
lendo o arquivo .assinatura */
    private void dadosArquivo(){
        String Dados = "";
        assinaturaDigital.dadosAssinatura        dados        =        new
assinaturaDigital.dadosAssinatura(jtfArquivo.getText());
        String nome  = dados.getAutor();
        String data  = dados.getData();
        String horas = dados.getHora();
        String path  = dados.getPath();
        String maqui = dados.getMaquina();
        String ip    = dados.getIp();
        if(!nome.trim().equalsIgnoreCase(""))   Dados  =  Dados  +"Autor=  "+
nome + "\n";
        if(!data.trim().equalsIgnoreCase(""))   Dados  =  Dados  +"Data=  "+
data + "\n";
        if(!horas.trim().equalsIgnoreCase("")) Dados  =  Dados  +"Horas=  "+
horas + "\n";
        if(!path.trim().equalsIgnoreCase(""))  Dados = Dados +"Arquivo= "+
path + "\n";
        if(!maqui.trim().equalsIgnoreCase("")) Dados = Dados +"Maquina= "+
maqui + "\n";
        if(!ip.trim().equalsIgnoreCase(""))    Dados = Dados +"Ip= "+ ip +
"\n";

        JOptionPane.showOptionDialog(this,          Dados,          "Dados",
JOptionPane.DEFAULT_OPTION,  JOptionPane.INFORMATION_MESSAGE,null,  options,
options[0]);

    }

    /** Obtem os dados da assinatura do documento assinado digitalmente
lendo do Banco de Dados */
    private void dadosBanco(){
        Assinatura.util.assinaBanco          banco          =          new
Assinatura.util.assinaBanco(true);
        String frase = banco.getAssinaDadosBanco(jtfArquivo.getText());
        banco.shutDown();
        String Dados = "";
        StringTokenizer dadosTkz = new StringTokenizer(frase, "|");
        String nome  = dadosTkz.nextToken();
        String data  = dadosTkz.nextToken();
        String horas = dadosTkz.nextToken();
        String path  = dadosTkz.nextToken();
        String maqui = dadosTkz.nextToken();
        String ip    = dadosTkz.nextToken();
        if(!nome.trim().equalsIgnoreCase(""))   Dados  =  Dados  +"Autor=  "+
nome + "\n";
        if(!data.trim().equalsIgnoreCase(""))   Dados  =  Dados  +"Data=  " +
data + "\n";
        if(!horas.trim().equalsIgnoreCase("")) Dados  =  Dados  +"Horas=  "+
horas + "\n";
        if(!path.trim().equalsIgnoreCase(""))  Dados = Dados +"Arquivo= "+
path + "\n";
```

```
        if(!maqui.trim().equalsIgnoreCase("")) Dados = Dados +"Maquina= "+
maqui + "\n";
        if(!ip.trim().equalsIgnoreCase(""))    Dados = Dados +"Ip= "+ ip +
"\n";

        JOptionPane.showOptionDialog(this,          Dados,          "Dados",
JOptionPane.DEFAULT_OPTION,  JOptionPane.INFORMATION_MESSAGE,null,  options,
options[0]);

    }

    /** Exit the Application */
    private void exitForm(java.awt.event.WindowEvent evt) {
        System.exit(0);
    }


    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        new Fr_Assinatura().show();
    }


    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JButton jbArquivo;
    private javax.swing.JButton jbAssinar;
    private javax.swing.JButton jbDados;
    private javax.swing.JButton jbGeraChaves;
    private javax.swing.JButton jbVerificar;
    private javax.swing.JCheckBox jcbBanco;
    private javax.swing.JTextField jtfArquivo;
    private javax.swing.JTextField jtfAutor;
    // End of variables declaration

}
```

- **Classe assinaBanco.java**

```
/*
 * assinaBanco.java
 *
 * Created on 4 de Dezembro de 2003, 11:31
 */

package Assinatura.util;

import java.io.*;
import javax.swing.*;

import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.util.*;
```

```
/**
 *
 * @author  Administrador
 */
public class assinaBanco {
    private static boolean DEBUG = true;

    private Connection conApJava;  // Conexão que será criada com o banco
    private JTable vTabela;    // Tabela onde os registros serão inseridos


    private javax.swing.JFileChooser jfcArquivo = new
javax.swing.JFileChooser();
    public Object[] options = { "OK" }; // botao mensagem dos framens de
avisos
    public Object[] btSimNao = { "Sim","Não" }; // botões mensagem dos
framens de avisos


    /** Cria uma nova instacia para assinaBanco */
    public assinaBanco() {
    }

    /** Cria uma nova instacia para assinaBanco
     * @param conectar Se for true instancia e conecta ao banco,
     * e se for false, instacia e desconecta do banco.
     */
    public assinaBanco(boolean conectar) {
        if(conectar){
            conectaBanco();
        }else{
            shutDown();
        }
    }


    /** Faz a Conecção com o Banco de Dados */
   public void conectaBanco() {
       String sUrl = "";
         sUrl =
"jdbc:interbase://127.0.0.1/D:/AssinaturaDigital/banco/ASSINATURA.GDB";
//         sUrl =
"jdbc:interbase://192.168.1.70/D:/AssinaturaDigital/banco/ASSINATURA.GDB";

    String vUsuario = "SYSDBA";
    String vSenha = "masterkey";

    // Carrega o Driver JDBC e abre a conexão com o banco
    try{
     Class.forName("interbase.interclient.Driver");
     conApJava = DriverManager.getConnection(sUrl, vUsuario, vSenha);
    }
     catch(ClassNotFoundException cnfex){ // Excessões geradas
      String Msg = "Impossível carregar o seguinte driver JDBC: " +
cnfex.getMessage() + "\nSe o mesmo estiver instalado favor verificar a
variável de ambiente CLASSPATH\n\n";
        String Titulo = " Erro ao carregar Driver JDBC";

      // cnfex.getMessage() => retorna interbase.interclient.Driver
      JOptionPane.showMessageDialog(null, Msg , Titulo,
JOptionPane.WARNING_MESSAGE);
```

```
        System.exit(0); // termina o programa
    }
    catch(SQLException sqlex){
      String Msg = "Não foi possível conectar o banco de dados ...";
        String Titulo = " Erro ao tentar conectar base de dados !";

      JOptionPane.showMessageDialog(null, Msg , Titulo,
JOptionPane.WARNING_MESSAGE);
        sqlex.printStackTrace();
    }
//     show();

   }


   /** Grava A Assinatura Digital no Banco de Dados
    * @param codigo Código Identificador Para a Gravação na Tabela
    * @param path Caminho Onde o Arquivo Gravado se Encontra
    * @param assina Assinatura do Arquivo
    * @param dados Dados Gerais da Assinatura
    */
   private void gravaBanco(int codigo, String path, String assina, String
dados){
        try {
            Statement stmt = conApJava.createStatement();
            String query = "INSERT INTO TBASSINATURA (COD_ASS, PATH_ASS,
ASSINATURA_ASS, DADOS_ASS) VALUES ("+codigo+",'" +
            path+"', '"+assina+"', '"+dados+"')";
            stmt.execute(query);
            stmt.close();
            stmt = null;
        } catch (Exception e) {
            e.printStackTrace();
        }

   }

   /** AlteraA Assinatura Digital no Banco de Dados
    * @param codigo Código Identificador Para a Gravação na Tabela
    * @param path Caminho Onde o Arquivo Gravado se Encontra
    * @param assina Assinatura do Arquivo
    * @param dados Dados Gerais da Assinatura
    */
   private void alteraBanco(int codigo, String path, String assina, String
dados){
        try {
            Statement stmt = conApJava.createStatement();
            String query = "UPDATE TBASSINATURA SET PATH_ASS='"+path+"',
ASSINATURA_ASS= '"+assina+
                           "', DADOS_ASS='"+dados+"' WHERE
COD_ASS="+codigo;
            if(DEBUG) System.out.println("query= "+query);
            stmt.execute(query);
            stmt.close();
            stmt = null;
        } catch (Exception e) {
            e.printStackTrace();
        }

   }
```

```
   /** Verifica se Já Existe o Arquivo Assinado, Caso não Exista Cria-se um
novo campo na tabela, Caso Exista, Altar o Campo na tabela
    * @param codigo Código Identificador Para a Gravação na Tabela
    * @param path Caminho Onde o Arquivo Gravado se Encontra
    * @param assina Assinatura do Arquivo
    * @param dados Dados Gerais da Assinatura
    */
   public void verificaTabela(String arquivo, String assinatura, String
dados){
     Statement smtp;
     ResultSet rsAssinatura;
     try{ // Gera a Query para o banco
       String consulta = "SELECT * FROM TBASSINATURA WHERE
PATH_ASS='"+arquivo+"'";
       smtp = conApJava.createStatement();
       rsAssinatura = smtp.executeQuery(consulta);
       boolean registro = rsAssinatura.next();
       if (!registro){
           if(DEBUG) System.out.println("Gravando!!!");
           smtp = conApJava.createStatement();
           rsAssinatura = smtp.executeQuery("SELECT COUNT(*) FROM
TBASSINATURA");
           boolean ok = rsAssinatura.next();
           int codigo = rsAssinatura.getInt("COUNT");
           gravaBanco(codigo+1, arquivo, assinatura, dados);
       }else{
           if(DEBUG) System.out.println("Alterando!!!");
           int codigo = rsAssinatura.getInt("COD_ASS");
           alteraBanco(codigo, arquivo, assinatura, dados);
       }
       smtp.close();
     }
     catch (SQLException sqlex){ // Erro ao executar a Query no banco
       sqlex.printStackTrace();
     }
   }


   /** Pega os Dados Gerais da Assinatura
    * @param arquivo Nome do Arquivo Assinado
    * @return Dados Gerais da Assinatura
    */
   public String getAssinaDadosBanco(String arquivo){
     Statement smtp;
     ResultSet rs;
     String dados = "";
     try{ // Gera a Query para o banco
       String consulta = "SELECT DADOS_ASS FROM TBASSINATURA WHERE
PATH_ASS='"+arquivo+"'";
       smtp = conApJava.createStatement();
       rs = smtp.executeQuery(consulta);
       boolean rr = rs.next();
       dados = rs.getString("DADOS_ASS");
       smtp.close();
     }
     catch (SQLException sqlex){ // Erro ao executar a Query no banco
       sqlex.printStackTrace();
     }
     return(dados);
   }
```

```
   /** Pega a Assinatura Digital do Banco de Dados
    * @param arquivo Nome do Arquivo Assinado
    * @return chavePublica Chave Publica do Arquivo Assinado
    */
   public byte[] getAssinaturaBanco(String arquivo){
     Statement smtp;
     ResultSet rs;
     byte[] assig = new byte[8192];
     boolean dados = false;
     int cont =0;
     try{ // Gera a Query para o banco
       String consulta = "SELECT ASSINATURA_ASS FROM TBASSINATURA WHERE
PATH_ASS='"+arquivo+"'";
       smtp = conApJava.createStatement();
       rs = smtp.executeQuery(consulta);
       boolean rr = rs.next();
       String assinatura = rs.getString("ASSINATURA_ASS");
       StringTokenizer assTkn = new StringTokenizer(assinatura, "^");
       while(assTkn.hasMoreElements()){
          int buffer = Integer.parseInt(assTkn.nextToken());
          assig[cont] = (byte)buffer;
          cont++;
       }
       System.out.println(cont);
       smtp.close();
     }
     catch (SQLException sqlex){ // Erro ao executar a Query no banco
       sqlex.printStackTrace();
     }
     return(assig);
   }

   /** Executa o ShutDown do Banco de Dados */
   public void shutDown(){
     try{ // Fecha a conexão co o banco de dados
       conApJava.close();
     }
     catch(SQLException sqlex){ // Não foi possível fechar a conexão
       System.err.println("Erro ao fechar conexão com o banco...");
       sqlex.printStackTrace();
     }
   }

}
```

Anexo C – Projetos de lei sobre assinatura digital

Projetos de lei que abordam o tema assinatura digital no Congresso Nacional na atualidade.

PROJETO DE LEI DO SENADO Nº 672, DE 1999

Dispõe sobre o comércio eletrônico

O CONGRESSO NACIONAL decreta:

CAPÍTULO I

DO COMÉRCIO ELETRÔNICO EM GERAL

Seção Única

Disposições Preliminares

Art. 1º – Esta Lei, que regula o comércio eletrônico em todo o território nacional, aplica-se a qualquer tipo de informação na forma de mensagem de dados usada no contexto de atividades comerciais.

Art. 2º – Considera-se, para os fins desta Lei:

I – *mensagem eletrônica* – a informação gerada, enviada, recebida ou arquivada eletronicamente, por meio óptico ou por meios similares, incluindo, entre outros, "intercâmbio eletrônico de dados" (EDI), correio eletrônico, telegrama, telex e fax;

II – *intercâmbio eletrônico de dados (EDI)* – a transferência eletrônica, de computador para computador, de informações estruturadas de acordo com um padrão estabelecido para tal fim;

III – *remetente* de uma mensagem eletrônica – a pessoa pela qual, ou em cujo nome, a mensagem eletrônica é enviada ou gerada antes de seu armazenamento, caso este se efetue;

IV – *destinatário* de uma mensagem eletrônica – a pessoa designada pelo remetente para receber a mensagem eletrônica;

V – *intermediário*, com respeito a uma mensagem eletrônica – a pessoa que, em nome de outra, envia, recebe ou armazena a mensagem eletrônica ou presta outros serviços com relação a essa mensagem;

VI – *sistema de informação* – é um sistema para geração, envio, recepção, armazenamento ou outra forma de processamento de mensagens eletrônicas.

Art. 3º – Na interpretação desta Lei, levar-se-á em consideração a necessidade de promover a uniformidade da aplicação de normas sobre o comércio eletrônico em nível internacional.

Art. 4º – Questões relativas a matérias regidas por esta Lei que nela não estejam expressamente disciplinadas serão solucionadas em conformidade, dentre outras, com os seguintes princípios gerais nos quais ela se inspira:

I – facilitar o comércio eletrônico interno e externo;

II – convalidar as operações efetuadas por meio das novas tecnologias da informação;

III – fomentar e estimular a aplicação de novas tecnologias da informação;

IV – promover a uniformidade do direito aplicável à matéria; e

V – apoiar as novas práticas comerciais.

CAPÍTULO II
DA APLICAÇÃO DE REQUISITOS LEGAIS ÀS MENSAGENS DE DADOS

Seção I
Do Reconhecimento Jurídico das Mensagens de Dados

Art. 5º – Serão reconhecidos os efeitos jurídicos, validade ou eficácia à informação sob a forma de mensagem eletrônica e àquela a que se faça remissão mediante a utilização dessa espécie de mensagem.

Seção II
Da Exigência de Informação Escrita e de Assinatura

Art. 6º – Quando a lei determinar que uma informação conste por escrito, este requisito considerar-se-á preenchido por uma mensagem eletrônica, desde que a informação nela contida seja acessível para consulta posterior.

Art. 7º – No caso de a lei exigir a assinatura de uma pessoa, este requisito considerar-se-á preenchido por uma mensagem eletrônica, desde que seja utilizado algum método para identificar a pessoa e indicar sua aprovação para a informação contida na mensagem.

Parágrafo único. O método utilizado deverá ser confiável e apropriado para os propósitos para os quais a mensagem for gerada ou comunicada, levando-se em consideração todas as circunstâncias do caso, inclusive qualquer acordo das partes a respeito.

Seção III
Da Exigência da Informação na Forma Original

Art. 8º – Quando a lei estabelecer que uma informação seja apresentada ou conservada na sua forma original, este requisito considerar-se-á preenchido por uma mensagem eletrônica, desde que:

I – haja garantia fidedigna de preservação da integridade da informação desde o momento da sua geração em sua forma final, como uma mensagem eletrônica ou de outra forma; e

II – a informação seja acessível à pessoa à qual ela deva ser apresentada.

Parágrafo único. Para os propósitos do inciso I:

I – presume-se íntegra a informação que permaneça completa e inalterada, salvo a adição de qualquer endosso das partes ou outra mudança que ocorra no curso normal da comunicação, armazenamento e exposição;

II – o grau de confiabilidade requerido será determinado à luz dos fins para os quais a informação for gerada, assim como de todas as circunstâncias do caso.

Seção IV
Da Exigência de Conservação das Mensagens de Dados

Art. 9º – Se a lei determinar que certos documentos, registros ou informações sejam conservados, este requisito considerar-se-á preenchido mediante a conservação de mensagens eletrônicas, desde que:

I – a informação que elas contenham seja acessível para consulta posterior;

II – as mensagens eletrônicas sejam conservadas no formato no qual tenham sido geradas, enviadas ou recebidas, ou num formato em que se possa demonstrar que representam exatamente as informações geradas, enviadas ou recebidas; e

III – se conserve, quando for o caso, toda informação que permita determinar a origem e o destino das mensagens e a data e hora em que foram enviadas ou recebidas.

Parágrafo único. A obrigação de conservar documentos, registros ou informações de acordo com o disposto neste artigo não se aplica àqueles dados que tenham por única finalidade facilitar o envio ou o recebimento da mensagem.

CAPÍTULO III
DA COMUNICAÇÃO DE MENSAGENS DE DADOS

Seção I
Da Alteração mediante Acordo

Art. 10 – Nas relações entre as partes que geram, enviam, recebem, armazenam ou, de qualquer outro modo, processam mensagens eletrônicas, as disposições deste capítulo poderão ser alteradas mediante comum acordo.

Seção II
Da Celebração e Validade dos Contratos

Art. 11 – Na celebração de um contrato, a oferta e sua aceitação podem ser expressas por mensagens eletrônicas.

Seção III
Do Reconhecimento das Mensagens de Dados

Art. 12 – Nas relações entre o remetente e o destinatário, se reconhecerá validade ou eficácia a uma declaração de vontade ou a qualquer outra declaração feita por meio de uma mensagem eletrônica.

Seção IV
Da Proveniência das Mensagens de Dados

Art. 13 – Nas relações entre o remetente e o destinatário, uma mensagem eletrônica será considerada proveniente do remetente quando ela for enviada:

I – pelo próprio remetente;
II – por uma pessoa autorizada a agir em nome do remetente;
III – por um sistema de informação programado pelo remetente, ou em seu nome, para operar automaticamente.

§ 1º O destinatário tem, ainda, direito a considerar uma mensagem eletrônica como proveniente do remetente:

I – quando aplicar corretamente um procedimento previamente aceito pelo remetente para verificar sua procedência; ou
II – quando a mensagem recebida resultar dos atos de uma pessoa cujas relações com o remetente ou com seus agentes lhe tenha dado acesso ao método usado pelo remetente para identificar as mensagens eletrônicas dele procedentes.

§ 2º O disposto no § 1º não se aplicará:

I – a partir do momento em que o destinatário for informado pelo remetente de que a mensagem eletrônica não é de sua emissão; ou
II – nos casos previstos no inciso II do § 1º, desde o momento em que o destinatário saiba ou devesse saber, se agisse com a devida diligência, que a mensagem eletrônica não procede do remetente.

Art. 14 – Presume-se que a mensagem eletrônica recebida corresponde àquela que o remetente pretendeu enviar, salvo quando o destinatário saiba ou devesse saber, se agisse com a devida diligência ou empregasse o procedimento pactuado, que a transmissão causou algum erro na mensagem.

Art. 15 – Presume-se que cada mensagem eletrônica recebida é uma mensagem distinta, salvo quando ela duplica uma outra e o destinatário saiba ou devesse saber, caso agisse com a devida diligência ou empregasse o procedimento pactuado, que se trata de duplicidade.

Seção V
Do Aviso de Recebimento

Art. 16 – Os arts. 17, 18 e 19 aplicam-se quando, antes ou durante o envio de uma mensagem eletrônica, ou por meio dessa mensagem, o remetente solicite ou pactue com o destinatário que este informe o seu recebimento.

Art. 17 – Se o remetente não pactuar com o destinatário que este informe o recebimento de uma mensagem de uma forma ou por um método particular, poderá ser informado o seu recebimento mediante qualquer comunicação ou ato do destinatário que baste para esse propósito.

Art. 18 – Quando o remetente declarar que os efeitos da mensagem eletrônica estão condicionados à recepção de um aviso de recebimento, a mensagem eletrônica considerar-se-á como não tendo sido enviada enquanto este não for recebido.

Art. 19 – No caso de o remetente não declarar que os efeitos da mensagem eletrônica estão condicionados à recepção de um aviso de recebimento e tal aviso não for recebido pelo remetente dentro do prazo estabelecido ou pactuado, ou, inexistindo este, o remetente poderá, em um prazo razoável:

I – notificar o destinatário declarando que nenhum aviso de recebimento foi recebido e estipulando um prazo adequado à efetivação dessa providência;
II – caso o aviso de recebimento não seja recebido dentro do prazo a que se refere o inciso I, o remetente poderá, notificando o destinatário, tratar a mensagem como se ela nunca tivesse sido enviada.

Art. 20 – A recepção, pelo remetente, do aviso de recebimento enviado pelo destinatário gera a presunção de que aquele tenha recebido a mensagem eletrônica pertinente.

Parágrafo único. A presunção a que se refere o *caput* não implica que a mensagem eletrônica corresponda à mensagem recebida.

Art. 21 – Quando o aviso de recebimento o declarar, presume-se que a mensagem eletrônica cumpre os requisitos técnicos pactuados ou previstos nas normas técnicas aplicáveis.

Seção VI
Do Tempo e Lugar de Despacho e Recebimento das Mensagens de Dados

Art. 22 – O envio de uma mensagem eletrônica ocorre quando esta entra em um sistema de informação alheio ao controle do remetente ou da pessoa que a envia em seu nome.

Art. 23 – O momento de recepção de uma mensagem eletrônica é determinado:

I – quando o destinatário designar um sistema de informação para o propósito de recebimento das mensagens eletrônicas:

a) pelo momento em que a mensagem eletrônica entrar no sistema de informação designado; ou

b) pelo momento em que a mensagem eletrônica for recuperada pelo destinatário, no caso de ela ser enviada para um sistema de informação do destinatário que não seja o sistema de informação designado.

II – quando o destinatário não designar um sistema de informação, pelo momento em que a mensagem eletrônica entrar no sistema de informação do destinatário.

Parágrafo único. Aplica-se o disposto neste artigo ainda que o sistema de informação esteja situado num lugar distinto daquele em que a mensagem eletrônica se considere recebida, de acordo com o disposto no artigo seguinte.

Art. 24 – Uma mensagem eletrônica se considera expedida e recebida nos locais onde o remetente e o destinatário têm seus estabelecimentos, respectivamente.

Parágrafo único. Para os fins do disposto neste artigo:

I – se o remetente ou o destinatário têm mais de um estabelecimento, considera-se aquele que guarda relação mais estreita com a transação subjacente ou, inexistindo esta, o seu estabelecimento principal;

II – se o remetente ou o destinatário não possuem estabelecimento, considera-se, para os fins deste artigo, o local de sua residência habitual.

CAPÍTULO IV
DISPOSIÇÕES FINAIS

Art. 25 – Esta Lei entra em vigor na data de sua publicação.

Art. 26 – O Poder Executivo regulamentará esta Lei no prazo de noventa dias, contados da data de sua publicação.

JUSTIFICAÇÃO

O avanço da tecnologia impõe a necessidade de adaptação do ordenamento jurídico às inovações introduzidas no cotidiano da sociedade. O uso cada vez mais acentuado da informática reclama seu disciplinamento jurídico, sob pena de surgirem questionamentos sobre a validade e eficácia da utilização desse instrumental.

O projeto que ora apresentamos a esta Casa trata do tema no que diz respeito ao comércio eletrônico.

O comércio eletrônico é uma realidade que se encontra em franca expansão e que reclama uma disciplina jurídica adequada, que se irradia por diversas áreas do direito, como, por exemplo, o direito das obrigações, o direito de propriedade intelectual e o direito tributário.

Com o presente projeto, pretendemos dar início às discussões legislativas acerca da utilização das mensagens eletrônicas nas atividades comerciais.

Em virtude da novidade do tema, estamos conscientes de que a proposição não é a palavra final sobre a matéria, havendo de colher aperfeiçoamentos ao longo de sua tramitação.

De forma resumida, é o seguinte o conteúdo da proposição:

a) não se negarão efeitos jurídicos às informações na forma de mensagem eletrônica;

b) quando a lei requerer que determinada informação conste por escrito, ou a assinatura de uma pessoa, ou que determinada informação seja apresentada ou conservada na sua forma original, ou ainda, que certos documentos, registros ou informações sejam conservados, estes requisitos poderão ser preenchidos por uma mensagem eletrônica, desde que observadas as condições que especifica;

c) na formação de um contrato, a oferta e sua aceitação podem ser expressas por mensagens eletrônicas, o mesmo prevalecendo para a declaração de vontade, cuja validade ou eficácia não poderá ser negada pelo fato de ser feita por meio de uma mensagem eletrônica;

d) são definidos os critérios a serem observados para que se indique a procedência, para que se informe o recebimento, e para que se estabeleçam o tempo e lugar de envio e recebimento de uma mensagem eletrônica.

A proposição é baseada na "Lei Modelo da UNCITRAL [Comissão das Nações Unidas para o Direito Comercial Internacional] sobre Comércio Eletrônico", de 1996, cuja elaboração tem por objetivo a sua incorporação ao direito interno dos diversos países, de forma a "promover a uniformidade no direito aplicável aos métodos de comunicação e armazenamento de informações substitutivos dos que utilizam papel", tendo em vista a globalização da economia, que tem provocado um enorme crescimento do comércio internacional, especialmente do comércio eletrônico.

São esses os motivos que nos levam a apresentar o presente projeto de lei, para cujo aperfeiçoamento e posterior aprovação contamos com o apoio dos ilustres pares.

Sala das Sessões,

Senador LÚCIO ALCÂNTARA

PROJETO DE LEI Nº 1483, DE 1999.

(Do Sr. Dr. Hélio)

Institui a fatura eletrônica e a assinatura digital nas transações de "comércio" eletrônico.

(CONSTITUA-SE COMISSÃO ESPECIAL NOS TERMOS DO ART. 34, II, DO RICD, PARA APRECIAR O PROJETO DE LEI Nº 1,483, DE 1999, A SER INTEGRADA PELAS SEGUINTES COMISSÕES: DE DEFESA DO CONSUMIDOR, MAIO AMBIENTE E MINORIAS; DE ECONOMIA, INDÚSTRIA E COMÉRCIO; CIÊNCIA E TECNOLOGIA, COMUNICAÇÃO E INFORMÁTICA; E DE CONSTITUIÇÃO E JUSTIÇA E DE REDAÇÃO)

O CONGRESSO NACIONAL decreta:

Art. 1º Fica instituída a fatura eletrônica assim como a assinatura digital, nas transações comerciais eletrônicas realizadas em todo território nacional.

Art. 2º A assinatura digital terá sua autenticação e reconhecimento certificado por órgão público que será regulamentado para este fim.

Parágrafo Único. Toda documentação eletrônica, bem como o cadastro de assinaturas digitais, deverão estar com seus registros disponíveis para avaliação e fiscalização dos órgãos federais responsáveis.

JUSTIFICAÇÃO

O avanço das tecnologias de informação caracterizado pelo "comércio eletrônico", introduziu um novo paradigma nas transações comerciais globais e na vida do cidadão comum.

Estima-se que as transações eletrônicas entre empresas, entre cidadãos e empresas, incorporam, transferências financeiras, os novos processos de teleeducação, de telemedicina, as certificações digitais, entre outros serviços.

Abre-se para os países detentores de tecnologia da informação um grande campo para a capacitação técnica, abertura de mercado, criação de novos empregos por valores agregados locais, e incremento de renda, desde que possua lei e normatização capaz de disciplinar o mercado brasileiro.

Sala das Sessões, 12 de agosto de 1999.

_____

DEPUTADO Dr. HÉLIO
PDT/SP

PROJETO DE LEI Nº 1589, DE 1999.

Atribui valor jurídico à digitação de documentos e dá outras providências. Dispõe sobre os documentos produzidos e os arquivados em meio eletrônico e dá outras providências

Autor: Luciano Pizzatto Dep. (PFL/PR)

PLS 00022/1996
PL 03173/1997

Atribui valor jurídico à digitação de documentos e dá outras providências.

Dispõe sobre os documentos produzidos e os arquivados em meio eletrônico e dá outras providências.

(EMENTA MODIFICADA)

PLS 00672/1999

"Dispõe sobre comércio eletrônico".

PLS 00152/1991
PL 04102/1993

" Regula a garantia constitucional da inviolabilidade de dados; define crimes praticados por meio de computador; altera a Lei nº 7.646, de 18 de dezembro de 1987, que dispõe sobre a proteção da propriedade intelectual de programas de computador e sua comercialização no País". (Ementa modificada).

"Define os crimes de uso indevido de computador e dá outras providências".

PL 01589/1999

" Dispõe sobre o comércio eletrônico, a validade jurídica do documento eletrônico e a assinatura digital, e dá outras providências"

APENSADO AO PL 1483/99

Confederação Nacional da Indústria
Coordenadoria de Assuntos Legislativos
Acompanhamento Legislativo

N.Câmara: PL 01589/1999
Data Apresentação: 30/08/99
N.Senado:
Origem: Câmara dos Deputados
N.Congresso:
Regime: Normal
N. Mensagem:
Poder Terminativo: Não

Espécie: Projeto de Lei Ordinária
Autor: Luciano Pizzatto Dep. (PFL/PR)

Ementa

" Dispõe sobre o comércio eletrônico, a validade jurídica do documento eletrônico e a assinatura digital, e dá outras providências"
APENSADO AO PL 1483/99

Íntegra
Data: 30/08/99

PROJETO NA CASA DE ORIGEM (CÂMARA DOS DEPUTADOS)
O Congresso Nacional decreta:

TÍTULO I - DEFINIÇÕES GERAIS
Capítulo I - Do âmbito de aplicação

Art. 1º - A presente lei regula o comércio eletrônico, a validade e o valor probante dos documentos eletrônicos, bem como a assinatura digital.

Capítulo II - Dos princípios gerais

Art. 2º - A interpretação da presente lei deve considerar o contexto in-ternacional do comércio eletrônico, o dinâmico progresso dos instrumentos tecnológicos, e a boa-fé das relações comerciais.
Parágrafo único - As questões relativas a matérias regidas pela presente lei, e que não estejam nela expressamente previstas, serão dirimidas de conformidade com os princípios gerais que dela decorrem.

TÍTULO II - COMÉRCIO ELETRÔNICO
Capítulo I - Da desnecessidade de autorização prévia

Art. 3º - O simples fato de ser realizada por meio eletrônico não sujeitará a oferta de bens, serviços e informações a qualquer tipo de autorização prévia.

Capítulo II - Das informações prévias

Art. 4º - A oferta de contratação eletrônica deve conter claras e inequívocas informações sobre:

a) nome do ofertante, e o número de sua inscrição no cadastro geral do Ministério da Fazenda, e ainda, em se tratando de serviço sujeito a regime de profissão regulamentada, o número de inscrição no órgão fiscalizador ou regulamentador;

b) endereço físico do estabelecimento;

c) identificação e endereço físico do armazenador;
d) meio pelo qual é possível contatar o ofertante, inclusive correio eletrônico;
e) o arquivamento do contrato eletrônico, pelo ofertante;
f) instruções para arquivamento do contrato eletrônico, pelo aceitante, bem como para sua recuperação, em caso de necessidade; e
g) os sistemas de segurança empregados na operação.

Capítulo III - Das informações privadas do destinatário

Art. 5º - O ofertante somente poderá solicitar do destinatário informa-ções de caráter privado necessárias à efetivação do negócio oferecido, devendo mantê-las em sigilo, salvo se prévia e expressamente autorizado a divulgá-las ou cedê-las pelo respectivo
titular.

§ 1º - A autorização de que trata o caput deste artigo constará em destaque, não podendo estar vinculada à aceitação do negócio.
§ 2º - Responde por perdas e danos o ofertante que solicitar, divulgar ou ceder informações em violação ao disposto neste artigo.

Capítulo IV - Da contratação eletrônica

Art. 6º - A oferta pública de bens, serviços ou informações à distância deve ser realizada em ambiente seguro, debidamente certificado.

Art. 7º - Os sistemas eletrônicos do ofertante deverão transmitir uma resposta eletrônica automática, transcrevendo a mensagem transmitida anteriormente pelo destinatário, e confirmando seu recebimento.

Art. 8º - O envio de oferta por mensagem eletrônica, sem prévio con-sentimento dos destinatários, deverá permitir a estes identificá-la como tal, sem que seja necessário tomarem conhecimento de seu conteúdo.

Capítulo V - Dos intermediários

Art. 9º - O intermediário que forneça serviços de conexão ou de trans-missão de informações, ao ofertante ou ao adquirente, não será responsável pelo conteúdo das informações transmitidas.

Art. 10 - O intermediário que forneça ao ofertante serviços de armaze-namento de arquivos e de sistemas necessários para operacionalizar a oferta eletrônica de bens, serviços ou informações, não será responsável pelo seu conteúdo, salvo, em ação
regressiva do ofertante, se:

a) deixou de atualizar, ou os seus sistemas automatizados deixaram de atualizar, as informações objeto da oferta, tendo o ofertante tomado as medidas adequadas para efetivar as atualizações, conforme instruções do próprio armazenador; ou

b) deixou de arquivar as informações, ou, tendo-as arquivado, foram elas destruídas ou modificadas, tendo o ofertante tomado as medidas adequadas para seu arquivamento, segundo parâmetros estabelecidos pelo armazenador.

Art. 11 - O intermediário, transmissor ou armazenador, não será obriga-do a vigiar ou fiscalizar o conteúdo das informações transmitidas ou armazenadas.

Parágrafo único - Responde civilmente por perdas e danos, e penalmente por co-autoria do delito praticado, o armazenador de informações que, tendo conhecimento inequívoco de que a oferta de bens, serviços ou informações constitui crime ou contravenção penal, deixar de promover sua imediata suspensão, ou interrupção de acesso por destinatários, competindo-lhe notificar, eletronicamente ou não, o ofertante, da medida adotada.

Art. 12 - O intermediário deverá guardar sigilo sobre as informações transmitidas, bem como sobre as armazenadas, que não se destinem ao conhecimento público.

Parágrafo único - Somente mediante ordem judicial poderá o intermediário dar acesso às informações acima referidas, sendo que as mesmas deverão ser mantidas, pelo respectivo juízo, em segredo de justiça.

Capítulo VI - Das normas de proteção e de defesa do consumidor

Art. 13 - Aplicam-se ao comércio eletrônico as normas de defesa e proteção do consumidor.

§ 1º - Os adquirentes de bens, de serviços e informações mediante contrato eletrônico poderão se utilizar da mesma via de comunicação adotada na contratação, para efetivar notificações e intimações extrajudiciais, a fim de exercerem direito consagrado nas normas de defesa do consumidor.

§ 2º - Deverão os ofertantes, no próprio espaço que serviu para oferecimento de bens, serviços e informações, disponibilizar área específica para fins do parágrafo anterior, de fácil identificação pelos consumidores, e que permita seu armazenamento, com data de transmissão, para fins de futura comprovação.

§ 3º - O prazo para atendimento de notificação ou intimação de que trata o parágrafo primeiro começa a fluir da data em que a respectiva mensagem esteja disponível para acesso pelo fornecedor.

§ 4º - Os sistemas eletrônicos do ofertante deverão expedir uma resposta eletrônica automática, incluindo a mensagem do remetente, confirmando o recebimento de quaisquer intimações, notificações, ou correios eletrônicos dos consumidores.

TÍTULO III - DOCUMENTOS ELETRÔNICOS

Capítulo I - Da eficácia jurídica dos documentos eletrônicos

Art. 14 - Considera-se original o documento eletrônico assinado pelo seu autor mediante sistema criptográfico de chave pública.

§1º - Considera-se cópia o documento eletrônico resultante da digitali-zação de documento físico, bem como a materialização física de documento eletrônico original.

§ 2º - Presumem-se conformes ao original as cópias mencionadas no parágrafo anterior, quando autenticadas pelo escrivão na forma dos arts. 33 e 34 desta lei.

§ 3º - A cópia não autenticada terá o mesmo valor probante do original, se a parte contra quem foi produzida não negar sua conformidade.

Art. 15 - As declarações constantes do documento eletrônico, digital-mente assinado, presumem-se verdadeiras em relação ao signatário, desde que a assinatura digital:

a) seja única e exclusiva para o documento assinado;
b) seja passível de verificação;
c) seja gerada sob o exclusivo controle do signatário;
d) esteja de tal modo ligada ao documento eletrônico que, em caso de posterior alteração deste, a assinatura seja invalidada; e
e) não tenha sido gerada posteriormente à expiração, revogação ou suspensão das chaves.

Art. 16 - A certificação da chave pública, feita pelo tabelião na forma do Capítulo II do Título IV desta lei, faz presumir sua autenticidade.

Art. 17 - A certificação de chave pública, feita por particular, prevista no Capítulo I do Título IV desta lei, é considerada uma declaração deste de que a chave pública certificada pertence ao titular indicado e não gera presunção de autenticidade perante terceiros.

Parágrafo único - Caso a chave pública certificada não seja autêntica, o particular, que não exerça a função de certificação de chaves como atividade econômica principal, ou de modo relacionado à sua atividade principal, somente responderá perante terceiros pelos danos causados quando agir com dolo ou fraude.

Art. 18 - A autenticidade da chave pública poderá ser provada por todos os meios de direito, vedada a prova exclusivamente testemunhal.

Art. 19 - Presume-se verdadeira, entre os signatários, a data do docu-mento eletrônico, sendo lícito, porém, a qualquer deles, provar o contrário por todos os meios de direito.

§ 1º - Após expirada ou revogada a chave de algum dos signatários, compete à parte a quem o documento beneficiar a prova de que a assinatura foi gerada anteriormente à expiração ou revogação.

§ 2º - Entre os signatários, para os fins do parágrafo anterior, ou em relação a terceiros, considerar-se-á datado o documento particular na data:

I - em que foi registrado;
II - da sua apresentação em repartição pública ou em juízo;

III - do ato ou fato que estabeleça, de modo certo, a anterioridade da formação do documento e respectivas assinaturas.

Art. 20 - Aplicam-se ao documento eletrônico as demais disposições legais relativas à prova documental, que não colidam com as normas deste Título.

Capítulo II - Da falsidade dos documentos eletrônicos

Art. 21 - Considera-se falso o documento eletrônico quando assinado com chaves fraudulentamente geradas em nome de outrem.

Art. 22 - O juiz apreciará livremente a fé que deva merecer o documen-to eletrônico, quando demonstrado ser possível alterá-lo sem invalidar a assinatura, gerar uma assinatura eletrônica idêntica à do titular da chave privada, derivar a chave privada a partir da chave pública, ou pairar razoável dúvida sobre a segurança do sistema criptográfico utilizado para gerar a assinatura.

Art. 23 - Havendo impugnação do documento eletrônico, incumbe o ônus da prova:

I - à parte que produziu o documento, quanto à autenticidade da chave pública e quanto à segurança do sistema criptográfico utilizado;
II - à parte contrária à que produziu o documento, quando alegar apropriação e uso da chave privada por terceiro, ou revogação ou suspensão das chaves.

Parágrafo único - Não sendo alegada questão técnica relevante, a ser dirimida por meio de perícia, poderá o juiz, ao apreciar a segurança do sistema criptográfico utilizado, valer-se de conhecimentos próprios, da experiência comum, ou de fatos notórios.

TÍTULO IV - CERTIFICADOS ELETRÔNICOS
Capítulo I - Dos certificados eletrônicos privados

Art. 24 - Os serviços prestados por entidades certificadoras privadas são de caráter comercial, essencialmente privados e não se confundem em seus efeitos com a atividade de certificação eletrônica por tabelião, prevista no Capítulo II deste Título.

Capítulo II - Dos certificados eletrônicos públicos
Seção I - Das certificações eletrônicas pelo tabelião

Art. 25 - O tabelião certificará a autenticidade de chaves públicas en-tregues pessoalmente pelo seu titular, debidamente identificado; o pedido de certificação será efetuado pelo requerente em ficha própria, em papel, por ele subscrita, onde constarão dados suficientes para identificação da chave pública, a ser arquivada em cartório.

§1º - O tabelião deverá entregar ao solicitante informações adequadas sobre o funcionamento das chaves pública e privada, sua validade e limitações, bem como sobre os procedimentos adequados para preservar a segurança das mesmas.

§ 2º - É defeso ao tabelião receber em depósito a chave privada, bem como solicitar informações pessoais do requerente, além das necessárias para desempenho de suas funções, devendo utilizá-las apenas para os propósitos da certificação.

Art. 26 - O certificado de autenticidade das chaves públicas deverá conter, no mínimo, as seguintes informações:

I - identificação e assinatura digital do tabelião;
II - data de emissão do certificado;
III - identificação da chave pública e do seu titular, caso o certificado não seja diretamente apensado àquela;
IV - elementos que permitam identificar o sistema criptografado utilizado;
V - nome do titular e poder de representação de quem solicitou a certificação, no caso de o titular ser pessoa jurídica.
Parágrafo único - Na falta de informação sobre o prazo de validade do certificado, este será de 2 (dois) anos, contados da data de emissão.

Seção 11 - Da revogação de certificados eletrônicos

Art. 27 - O tabelião deverá revogar um certificado eletrônico:

a) a pedido do titular da chave de assinatura ou de seu representante;
b) de ofício ou por determinação do Poder Judiciário, caso se verifique que o certificado foi expedido baseado em informações falsas; e
c) se tiver encerrado suas atividades, sem que tenha sido sucedido por outro tabelião.

§ 1º- A revogação deve indicar a data a partir da qual será aplicada.
§ 2º - Não se admite revogação retroativa, salvo nas hipóteses dos parágrafos 3º e 4º do art. 28.

Art. 28 - O titular das chaves é abrigado a adotar as medidas necessárias para manter a confidencialidade da chave privada, devendo revogá-la de pronto, em caso de comprometimento de sua segurança.

§ 1º - A revogação da chave pública certificada deverá ser feita perante o tabelião que emitiu o certificado; se a chave revogada contiver certificados de autenticidade de vários oficiais, a revogação poderá ser feita perante qualquer deles, ao qual competirá informar os demais, de imediato.
§ 2º - A revogação da chave pública somente poderá ser solicitada pelo seu titular ou por procurador expressamente autorizado.
§ 3º - Pairando dúvida sobre a legitimidade do requerente, ou não ha-vendo meios de demonstrá-la em tempo hábil, o tabelião suspenderá provisoriamente, por até trinta dias, a eficácia da chave pública, notificando imediatamente o seu titular, podendo, para tanto, utilizar-se de mensagem eletrônica; revogada a chave dentro deste prazo, os efeitos da revogação retroagirão à data da suspensão.

§ 4º - Havendo mera dúvida quanto à segurança da chave privada, é lícito ao titular pedir a suspensão dos certificados por até trinta dias, aplicando-se o disposto na parte final do parágrafo anterior.

Art. 29 - O tabelião deverá manter serviço de informação, em tempo real e mediante acesso eletrônico remoto, sobre as chaves por ele certificadas, tornando-as acessíveis ao público, fazendo-se menção às que tenham sido revogadas.

Art. 30 - O tabelião somente poderá certificar chaves geradas por sis-tema ou programa de computador que tenha recebido parecer técnico favorável a respeito de sua segurança e confiabilidade, emitido pelo Ministério da Ciência e Tecnologia.

Seção III - Do encerramento das atividades de certificação

Art. 31 - Caso encerre as atividades de certificação eletrônica, o tabeli-ão deverá assegurar que os certificados emitidos sejam transferidos para outro tabelião, ou sejam bloqueados.

Art. 32 - O tabelião deverá transferir as documentações referidas nos arts. 25 e 40 desta lei, ao tabelião que lhe suceder, ou, caso não haja sucessão, ao Poder Judiciário.

Seção IV - Da autenticação eletrônica

Art. 33 - A assinatura digital do tabelião, lançada em cópia eletrônica de documento físico original, tem o valor de autenticação.

Art. 34 - A autenticação de cópia física de documento eletrônico origi-nal conterá:

a) o nome dos que nele apuseram assinatura digital;
b) os identificadores das chaves públicas utilizadas para conferência das assinaturas e respectivas certificações que contiverem;
c) a data das assinaturas;
d) a declaração de que a cópia impressa confere com o original eletrônico e de que as assinaturas digitais foram conferidas pelo escrivão com o uso das chaves públicas acima indicadas;
e) data e assinatura do escrivão.

Seção V - Da responsabilidade dos tabeliães

Art. 35 - O tabelião é responsável civilmente pelos danos diretos e indi-retos sofridos pelos titulares dos certificados e quaisquer terceiros, em conseqüência do descumprimento, por si próprios, seus prepostos ou substitutos que indicarem, das obrigações decorrentes do presente diploma e sua regulamentação, que indicarem, das obrigações decorrentes do presente diploma e sua regulamentação.

Seção VI - Dos Registros Eletrônicos

Art. 36 - O Registro de Título e Documentos fica autorizado a proceder à transcrição e ao registro de documentos eletrônicos particulares, para os fins previstos na Lei nº 6.015, de 31 de dezembro de 1973.

Parágrafo único - Poderá o Poder Judiciário autorizar o uso de documentos eletrônicos em atividades notariais e de registro não previstas ex-pressamente na presente lei, adotando a regulamentação adequada, considerando inclusive as questões de segurança envolvidas.

TÍTULO V - AUTORIDADES COMPETENTES

Capítulo I - Do Poder Judiciário

Art. 37 - Compete ao Poder Judiciário:

a) autorizar os tabeliães a exercerem atividade de certificação eletrônica;
b) regulamentar o exercício das atividades de certificação, obedecidas as disposições desta lei;
c) fiscalizar o cumprimento, pelos tabeliães, do disposto nesta lei e nas normas por ele adotadas, quanto ao exercício de suas funções; e
d) impor as penalidades administrativas cabíveis, obedecido o processo legal, e independente das responsabilidades civis e penais dos tabeliães e seus oficiais.
Parágrafo único. Não será deferida autorização ao exercício da ativida-de de certificação eletrônica a tabelião que não apresentar
parecer técnico favorável emitido pelo Ministério da Ciência e Tecnologia.

Capítulo II - Do Ministério da Ciência e Tecnologia

Art. 38 - Compete ao Ministério de Ciência e Tecnologia:

a) regulamentar os aspectos técnicos do exercício de atividade de certificação eletrônica pelos tabeliães, dispondo inclusive sobre os elementos que devam ser observados em seus planos de segurança;
b) emitir parecer técnico sobre solicitação de tabelião para o exercício de atividade de certificação eletrônico; e
c) emitir os certificados para chaves de assinatura a serem utilizadas pelos tabeliães para firmarem certificados, devendo manter constantemente acessíveis ao público os certificados que tenha emitido, através de conexão por instrumentos de telecomunicações.

§ 1º - O Ministério da Ciência e Tecnologia revisará a cada 2 (dois) anos o regulamento técnico da certificação eletrônica, previsto na alínea a deste artigo, de forma a mantê-lo atualizado de acordo com os avanços da tecnologia.
§ 2º - Não será emitido parecer técnico favorável ao solicitante que:

a) não apresentar conhecimento ou as condições técnicas necessárias para o exercício de suas atividades;

b) não apresentar plano de segurança, ou, apresentando-o, for ele indeferido, ou ainda, caso seja constatado que o plano por ele proposto não está adequadamente implantado em suas dependências e sistemas.

Art. 39 - Deverá o Ministério da Ciência e Tecnologia promover fiscalização em periodicidade adequada, quanto ao cumprimento, pelos tabeliães, das normas técnicas por ele adotadas.

Parágrafo único - Apurando a fiscalização de que trata este artigo qual-quer irregularidade no cumprimento das normas técnicas, deverá notificar o tabelião para apresentar defesa no prazo máximo de 5 (cinco) dias, bem como emitir, a propósito da defesa apresentada, manifestação fundamentada, em igual prazo, encaminhando os autos para o Poder Judiciário decidir.

Art. 40 - O tabelião deverá:

a) documentar os sistemas que emprega na certificação, e as medidas constantes de seu plano de segurança, permitindo acessoa essa documentação pela fiscalização do Ministério de Ciência e Tecnologia; e

d) documentar os certificados expedidos, vigentes, esgotados e revogados, permitindo acesso a essa documentação pela

fiscalização do Poder Judiciário.

TÍTULO VI - SANÇÕES ADMINISTRATIVAS

Art. 41 - As infrações às normas estabelecidas nos Títulos IV e V desta lei, independente das sanções de natureza penal, e reparação de danos que causarem, sujeitam os tabeliães às seguintes penalidades:

I - multa, de RS 10. 000, 00 (dez mil reais) a R$ 1. 000. 000, 00 (um milhão de reais);

II - suspensão de certificado;

III - cancelamento de certificado;

IV - suspensão da autorização para exercício de atividade de certificação eletrônica;

V - cassação da autorização para exercício de atividade de certificação eletrônica;

V - cassação de licença de funcionamento.

Art. 42 - As sanções estabelecidas no artigo anterior serão aplicadas pelo Poder Judiciário, considerando-se a gravidade dainfração, vantagem auferida, capacidade econômica, e eventual reincidência.

Parágrafo único - As penas previstas nos incisos II e IV poderão ser impostas por medida cautelar antecedente ou incidente de procedimento administrativo.

TÍTULO VII - SANÇÕES PENAIS

Art. 43 - Equipara-se ao crime de falsificação de papéis públicos, sujei-tando-se às penas do art. 293 do Código Penal, a falsificação, com fabricação ou alteração, de certificado eletrônico público.

Parágrafo único - Incorre na mesma pena de crime de falsificação de papéis públicos quem utilizar certificado eletrônico público falsificado.

Art. 44 - Equipara-se ao crime de falsificação de documento público, sujeitando-se às penas previstas no art. 297 do Código Penal, a falsificação, no todo ou em parte, de documento eletrônico público, ou alteração de documento eletrônico público verdadeiro.

Parágrafo único - Se o agente é funcionário público, e comete o crime prevalecendo-se do cargo, aplica-se o disposto no § 1º do art. 297 do Código Penal.

Art. 45 - Equipara-se ao crime de falsidade de documento particular, sujeitando-se às penas do art. 298 do Código Penal, a falsificação, no todo ou em parte, de documento eletrônico particular, ou alteração de documento eletrônico particular verdadeiro.

Art. 46 - Equipara-se ao crime de falsidade ideológica, sujeitando-se às penas do art. 299 do Código Penal, a omissão, em documento eletrônico público ou particular, de declaração que dele devia constar, ou a inserção ou fazer com que se efetue inserção, de declaração falsa ou diversa da que devia ser escrita, com o fim de prejudicar direito, criar obrigação ou alterar a verdade sobre fato juridicamente relevante.

Parágrafo único - Se o agente é funcionário público, e comete o crime prevalecendo-se do cargo, aplica-se o disposto no parágrafo único do art. 299 do Código Penal.

Art. 47 - Equipara-se ao crime de falso reconhecimento de firma, sujei-tando-se às penas do art. 300 do Código Penal, o reconhecimento, como verdadeiro, no exercício de função pública, de assinatura eletrônica, que não o seja.

Art. 48 - Equipara-se ao crime de supressão de documento, sujeitando-se às penas do art. 305 do Código Penal, a destruição, supressão ou ocultação, em benefício próprio ou de outrem, de documento eletrônico público ou particular verdadeiro, de que não se poderia dispor.

Art. 49 - Equipara-se ao crime de extravio, sonegação ou inutilização de documento, sujeitando-se às penas previstas no art. 314 do Código Penal, a extravio de qualquer documento eletrônico, de que se tem a guarda em razão do cargo; ou sua sonegação ou inutilização, total ou parcial.

TÍTULO VIII - DISPOSIÇÕES GERAIS

Art. 50 - As certificações estrangeiras de assinaturas digitais terão o mesmo valor jurídico das expedidas no país, desde que entidade certificadora esteja sediada e seja devidamente reconhecida, em país signatário de acordos internacionais dos quais sejaparte o Brasil, relativos ao reconhecimento jurídico daqueles certificados.

Parágrafo único - O Ministério da Ciência e Tecnologia fará publicar os nomes das entidades certificadoras estrangeiras que atendam aos requisitos determinados neste artigo.

Art. 51 - Para a solução de litígios de matérias objeto desta lei poderá ser empregado sistema de arbitragem, obedecidos os parâmetros da Lei nº 9.037, de 23 de setembro de 1996, dispensada a obrigação decretada no § 2º de seu art. 4º, devendo, entretanto, efetivar-se destacadamente a contratação eletrônica da cláusula compromissória.

TÍTULO IX - DISPOSIÇÕES FINAIS

Art. 52 - O Poder Executivo regulamentará a presente lei no prazo de 30 dias, após o qual deverão o Ministério da Ciência e Tecnologia e o Poder Judiciário, no prazo de 60 dias, baixar as normas necessárias para o exercício das atribuições conferidas pela presente lei.

JUSTIFICAÇÃO

1. Os avanços tecnológicos têm causado forte impacto sobre as mais diversas áreas do conhecimento e das relações humanas.

O comércio eletrônico representa um dos exemplos mais significativos dessa verdadeira revolução social.

2. O direito, por sua vez, tem por uma de suas principais características o hiato temporal existente entre o conhecimento das mudanças sociais, sua compreensão, as tentativas iniciais de tratá-las à luz de conceitos tradicionais e, finalmente, a adoção de princípios próprios para regular as relações que delas resultam.

Essa característica, que tem o grande mérito de assegurar a segurança jurídica mesmo nas grandes revoluções sociais, encontra, porém, na velocidade com que a tecnologia as têm causado, também seu impacto, requerendo seja menor o tempo necesario para adoção de disciplina para as novas relações sociais.

3. Diversos países já adotaram leis especiais tratando das transações eletrônicas, especialmente no que se refere, à questão do documento eletrônico e da assinatura digital.

4. A primeira lei dispondo sobre essas questões foi promulgada pelo Estado de Utah, denominada Digital Signature Act, ou Lei da Assinatura Digital. Hoje, a maioria dos Estados norte-americanos já dispõe de leis tratando, com maior ou menor abrangência, dessa matéria, sendo hoje, a grande preocupação harmonizar em nível federal essas legislações.

5. Na Europa, também, diversos países já adotaram leis específicas dispondo sobre essas questões: Itália, Alemanha, e mais recentemente Portugal, já promulgaram leis próprias. E já há, também, no âmbito da Comunidade Européia, a

preocupação de definir parâmetros a serem adotados por todos os países que a compõe, de forma a permitir harmonização entre essas diferentes leis nacionais.

6. Na América Latina já existem igualmente leis dispondo sobre documentos eletrônicos e assinatura digital.

A Argentina, por exemplo, teve no Decreto nº 427, de 16 de abril de 1998, o marco inicial na regulamentação da assinatura digital, embora restrita ao âmbito da administração pública. Tem a Argentina, atualmente, anteprojeto de lei apresentado pela Comissão Redatora nomeada pelo Ministério da Justiça.

O Uruguai, o marco para validade do documento eletrônico foi a promulgação da Lei nº 16.002, de 25 de novembro de 1988, posteriormente alterada pela Lei nº 16.736, de 5 de janeiro de 1996, universalizando a origem e o destino do documento eletrônico, para fins de reconhecimento legal, que antes tinha seu reconhecimento limitado às correspondências entre órgãos governamentais.

7. Ao lado da preocupação em assegurar validade jurídica ao documento eletrônico e à assinatura digital, surgiu, em meados desta década, outra preocupação: a de disciplinar o próprio comércio eletrônico.

8. Em 1996, a UNCITRAL adotou Lei Modelo sobre Comércio Eletrônico, propondo as principais normas a serem adotadas nas legislações nacionais, visando a criar ambiente internacional para o desenvolvimento dessa nova modalidade de negócios.

Em 1º de julho de 1997, o Presidente dos Estados Unidos, Bill Clinton, propôs uma série de linhas mestras a serem adotadas pelos países, quer no âmbito.

No mesmo período ocorreu a "Global Information Networks: Realizing the Potencial", em Bona, que resultou em recomendações sobre o comércio eletrônico no âmbito da Comunidade Européia e da cooperação internacional.

Desses movimentos nasceu, no final daquele ano, a declaração conjunta sobre comércio eletrônico, firmada pelos presidentes dos Estados Unidos e da Comunidade Européia.

9. Ainda no âmbito da Comunidade Européia, encontra-se em final de tramitação proposta de diretiva do Parlamento Europeu e do Conselho, visando a definir um quadro de assinaturas eletrônicas.

Verificou-se que as legislações nacionais, e mesmo as estaduais, no caso dos Estados Unidos, contemplam solução única para ambos os problemas: a adoção da criptografia assimétrica que, significando enorme avanço em relação à criptografia tradicional, simétrica, é composta por duas chaves, uma privada, de conhecimento exclusivo de seu titular, e uma pública, de conhecimento público.

10. O emprego dessa técnica deve considerar a existência de uma terceira parte: a autoridade certificadora, ou entidade certificante, a quem compete certificar a titularidade da chave pública, dando credibilidade à assinatura e ao documento eletrônicos.

11. Na disciplina dessas entidades, foi necessário considerar o disposto no art. 236 da Constituição do Brasil, que dispõe sobre os serviços notariais e de registro, exercidos em caráter privado mas por delegação do Poder Público, e definidos, pelo art. 1º da Lei nº 8.935, de 18 de novembro de 1994, que regulamentou referido dispositivo constitucional, como aqueles destinados a garantir a publicidade, autenticidade, segurança e eficácia dos atos jurídicos - exatamente o que a certificação visa em relação à assinatura e ao documento eletrônicos.

12. Dividiu-se, assim, a atividade de certificação, em dois grupos distintos, com eficácias diferentes: as certidões eletrônicas por entidades privadas, de caráter comercial, essencialmente privado; e as certidões eletrônicas por tabeliães, de

caráter público, e que geram presunção de autenticidade do documento ou da assinatura eletrônica.

13. Com essa disciplina distinta, se legitima a atuação das entidades privadas de certificação, importantes, mas que não têm fé pública, restringida esta aos tabeliães.

14. Dessa regra decorrerá toda a disciplina proposta no anteprojeto, em relação à validade jurídica do documento digital.

15. Destaque-se também que, em relação à atividade pública de certificação, realizada pelos tabeliães, decidiu-se propor no anteprojeto duas autoridades distintas, no controle daquela atividade:

a) o Poder Judiciário, a quem, nos termos do art. 236 da Constituição do Brasil, compete sua fiscalização; e

b) o Ministério da Ciência e Tecnologia, que cumprirá papel das definições técnicas, inclusive quanto à segurança adequada para o uso da tecnologia de certificações.

16. É também importante destacar que o anteprojeto partiu do princípio de que os conceitos tradicionais não devem ser pura e simplesmente afastados, mas sim ajustados à realidade do comércio eletrônico, dando segurança maior às partes, inclusive no que diz respeito aos futuros pronunciamentos do próprio Poder Judiciário.

Assim, o projeto adotou a técnica de não pretender conceituar os novos institutos, nem criar novos tipos jurídicos, preferindo inclusive manter o estilo de redação dos dispositivos que já dispõem sobre aspectos jurídicos do documento eletrônico, seja no âmbito civil, seja na tipificação penal, de forma a permitir melhor compreensão por parte dos operadores do direito.

17. Finalmente, destaque-se também que o anteprojeto, levando ainda em consideração que o comércio eletrônico tem, como das principais características, a transnacionalidade, propõe tenham as certificações estrangeiras a mesma eficácia das certificações nacionais, desde que a entidade certificadora tenha sede em país signatário de acordos internacionais dos quais seja parte o Brasil, relativos ao reconhecimento jurídico dos certificados eletrônicos.